



---

## wxErlang

Copyright © 2009-2010 Ericsson AB. All Rights Reserved.  
wxErlang 0.98.7  
September 16 2010

---

**Copyright © 2009-2010 Ericsson AB. All Rights Reserved.**

The contents of this file are subject to the Erlang Public License, Version 1.1, (the "License"); you may not use this file except in compliance with the License. You should have received a copy of the Erlang Public License along with this software. If not, it can be retrieved online at <http://www.erlang.org/>. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. Ericsson AB. All Rights Reserved..

**September 16 2010**



# 1 User's Guide

---

The *wxErlang* application is an api for writing graphical user interfaces with wxWidgets.

## 1.1 wx the erlang binding of wxWidgets

The *wx* application is an erlang binding of *wxWidgets*. This document describes the erlang mapping to wxWidgets and its implementation. It is not a complete users guide to wxWidgets. If you need that, you will have to read the wxWidgets documentation instead. *wx* tries to keep a one-to-one mapping with the original api so that the original documentation and examples shall be as easy as possible to use.

wxErlang examples and test suite can be found in the erlang src release. They can also provide some help on how to use the api.

This is currently a very brief introduction to *wx*. The application is still under development, which means the interface may change, and the test suite currently have a poor coverage ratio.

### 1.1.1 Contents

- *Introduction*
- *Multiple processes and memory handling*
- *Event Handling*
- *Acknowledgments*

### 1.1.2 Introduction

The original *wxWidgets* is an object-oriented (C++) api and that is reflected in the erlang mapping. In most cases each class in wxWidgets is represented as a module in erlang. This gives the *wx* application a huge interface, spread over several modules, and it all starts with the *wx* module. The *wx* module contains functions to create and destroy the gui, i.e. `wx:new/0`, `wx:destroy/0`, and some other useful functions.

Objects or object references in *wx* should be seen as erlang processes rather than erlang terms. When you operate on them they can change state, e.g. they are not functional objects as erlang terms are. Each object has a type or rather a class, which is manipulated with the corresponding module or by sub-classes of that object. Type checking is done so that a module only operates on its objects or inherited classes.

An object is created with *new* and destroyed with *destroy*. Most functions in the classes are named the same as their C++ counterpart, except that for convenience, in erlang they start with a lowercase letter and the first argument is the object reference. Optional arguments are last and expressed as tagged tuples in any order.

For example the *wxWindow* C++ class is implemented in the *wxWindow* erlang module and the member `wxWindow::CenterOnParent` is thus `wxWindow:centerOnParent`. The following C++ code:

```
wxWindow MyWin = new wxWindo();
MyWin.CenterOnParent(wxVERTICAL);
...
delete MyWin;
```

would in erlang look like:

```
MyWin = wxWindow:new(),
wxWindow:centerOnParent(MyWin, [{dir,?wxVERTICAL}]),
...
wxWindow:destroy(MyWin),
```

When you are reading wxWidgets documentation or the examples, you will notice that some of the most basic classes are missing in wx, they are directly mapped to corresponding erlang terms:

*wxPoint* is represented by {Xcoord,Ycoord}  
*wxSize* is represented by {Width,Height}  
*wxRect* is represented by {Xcoord,Ycoord,Width,Height}  
*wxColour* is represented by {Red,Green,Blue[,Alpha]}  
*wxPoint* is represented by {Xcoord,Ycoord}  
*wxString* is represented by *unicode:charlist()*  
*wxGBPosition* is represented by {Row,Column}  
*wxGBSpan* is represented by {RowSpan,ColumnSPAN}  
*wxGridCellCoords* is represented by {Row,Column}

In the places where the erlang api differs from the original one it should be obvious from the erlang documentation which representation has been used. E.g. the C++ arrays and/or lists are sometimes represented as erlang lists and sometimes as tuples.

Colours are represented with {Red,Green,Blue[,Alpha]}, the Alpha value is optional when used as an argument to functions, but it will always be returned from wx functions.

Defines, enumerations and global variables exists in `wx.hrl` as `defines`. Most of these defines are constants but not all. Some are platform dependent and therefore the global variables must be instantiated during runtime. These will be acquired from the driver with a call, so not all defines can be used in matching statements. Class local enumerations will be prefixed with the class name and a underscore as in `ClassName_Enum`.

Additionally some global functions, i.e. non-class functions, exist in the `wx_misc` module.

*wxErlang* is implemented as a (threaded) driver and a rather direct interface to the C++ api, with the drawback that if the erlang programmer does an error, it might crash the emulator.

Since the driver is threaded it requires a *smp* enabled emulator, that provides a thread safe interface to the driver.

### 1.1.3 Multiple processes and memory handling

The intention is that each erlang application calls `wx:new()` once to setup it's gui which creates an environment and a memory mapping. To be able to use wx from several processes in your application, you must share the environment. You can get the active environment with `wx:get_env/0` and set it in the new processes with `wx:set_env/1`. Two processes or applications which have both called `wx:new()` will not be able use each others objects.

```
wx:new(),
MyWin = wxFrame:new(wx:null(), 42, "Example", []),
Env = wx:get_env(),
spawn(fun() ->
    wx:set_env(Env),
    %% Here you can do wx calls from your helper process.
    ...
end),
...
```

When `wx:destroy/0` is invoked or when all processes in the application have died, the memory is deleted and all windows created by that application are closed.

## 1.1 wx the erlang binding of wxWidgets

---

The *wx* application never cleans or garbage collects memory as long as the user application is alive. Most of the objects are deleted when a window is closed, or at least all the objects which have a parent argument that is non null. By using `wxCCLASS:destroy/1` when possible you can avoid an increasing memory usage. This is especially important when *wxWidgets* assumes or recommends that you (or rather the C++ programmer) have allocated the object on the stack since that will never be done in the erlang binding. For example `wxDC` class or its sub-classes or `wxSizerFlags`.

Currently the dialogs show modal function freezes *wxWidgets* until the dialog is closed. That is intended but in erlang where you can have several gui applications running at the same time it causes trouble. This will hopefully be fixed in future *wxWidgets* releases.

### 1.1.4 Event Handling

Event handling in *wx* differs most the from the original api. You must specify every event you want to handle in *wxWidgets*, that is the same in the erlang binding but can you choose to receive the events as messages or handle them with callback funs.

Otherwise the event subscription is handled as *wxWidgets* dynamic event-handler connection. You subscribe to events of a certain type from objects with an *ID* or within a range of *ID*:s. The callback fun is optional, if not supplied the event will be sent to the process that called `connect/2`. Thus, a handler is a callback fun or a process which will receive an event message.

Events are handled in order from bottom to top, in the widgets hierarchy, by the last subscribed handler first. Depending on if `wxEvent:skip()` is called the event will be handled by the other handler(s) afterwards. Most of the events have default event handler(s) installed.

Message events looks like `#wx{id=integer(), obj=wx:wxObject(), userData=term(), event=Rec }`. The *id* is the identifier of the object that received the event. The *obj* field contains the object that you used `connect` on. The *userData* field contains a user supplied term, this is an option to `connect`. And the *event* field contains a record with event type dependent information. The first element in the event record is always the type you subscribed to. For example if you subscribed to `key_up` events you will receive the `#wx{event=Event }` where *Event* will be a `wxKey` event record where `Event#wxKey.type = key_up`.

In *wxWidgets* the developer have to call `wxEvent:skip()` if he wants the event to be processed by other handlers. You can do the same in *wx* if you use callbacks. If you want the event as messages you just don't supply a callback and you can set the *skip* option in `connect` call to true or false, the default it is false. True means that you get the message but let the subsequent handlers also handle the event. If you want to change this behavior dynamically you must use callbacks and call `wxEvent:skip()`.

Callback event handling is done by using the optional callback *fun/2* when attaching the handler. The `fun(#wx{}),wxObject()` must take two arguments where the first is the same as with message events described above and the second is an object reference to the actual event object. With the event object you can call `wxEvent:skip()` and access all the data. When using callbacks you must call `wxEvent:skip()` by yourself if you want any of the events to be forwarded to the following handlers. The actual event objects are deleted after the *fun* returns.

The callbacks are always invoked by another process and have exclusive usage of the gui when invoked. This means that a callback fun can not use the process dictionary and should not make calls to other processes. Calls to another process inside a callback fun may cause a deadlock if the other process is waiting on completion of his call to the gui.

### 1.1.5 Acknowledgments

Mats-Ola Persson wrote the initial *wxWidgets* binding as part of his master thesis. The current version is a total re-write but many ideas have been reused. The reason for the re-write was mostly due to the limited requirements he had been given by us.

Also thanks to the *wxWidgets* team that develops and supports it so we have something to use.

## 2 Reference Manual

---

The *wxErlang* application is an api for writing graphical user interfaces with wxWidgets.

## WX

---

Erlang module

A port of **wxWidgets**.

This is the base api of **wxWidgets**. This module contains functions for starting and stopping the wx-server, as well as other utility functions.

wxWidgets is object oriented, and not functional. Thus, in wxErlang a module represents a class, and the object created by this class has an own type, wxCLASS(). This module represents the base class, and all other wxMODULE's are sub-classes of this class.

Objects of a class are created with wxCLASS:new(...) and destroyed with wxCLASS:destroy(). Member functions are called with wxCLASS:member(Object, ...) instead of as in C++ Object.member(...).

Sub class modules inherit (non static) functions from their parents. The inherited functions are not documented in the sub-classes.

This erlang port of wxWidgets tries to be a one-to-one mapping with the original wxWidgets library. Some things are different though, as the optional arguments use property lists and can be in any order. The main difference is the event handling which is different from the original library. See *wxEvtHandler*.

The following classes are implemented directly as erlang types:

wxPoint={x,y}, wxSize={w,h}, wxRect={x,y,w,h}, wxColour={r,g,b [a]}, wxString=*unicode:charlist()*,  
wxGBPosition={r,c}, wxGBSpan={rs,cs}, wxGridCellCoords={r,c}.

wxWidgets uses a process specific environment, which is created by *wx:new/0*. To be able to use the environment from other processes, call *get\_env/0* to retrieve the environment and *set\_env/1* to assign the environment in the other process.

Global (classless) functions are located in the wx\_misc module.

## DATA TYPES

`colour()`

A 3 or 4 tuple: {R,G,B,A} or as argument {R,G,B} is also accepted where each colour channel is an integer between 0-255.

`datetime()`

{{Year,Month,Day}, {Hour,Minute,Second}} in local timezone.

`mouseState()`

See #wxMouseState{} defined in wx.hrl

`wxObject()`

Opaque object

`wx_env()`

Wx process environment

`wx_mem()`

Wx memory area



## Exports

**new()** -> wxObject()

Starts a wx server.

**new(Options::[Option])** -> wxObject()

Starts a wx server. Option may be {debug, Level}, see debug/1.

**destroy()** -> ok

Stops a wx server.

**get\_env()** -> wx\_env()

Gets this process's current wx environment. Can be sent to other processes to allow them use this process wx environment.

*See also: set\_env/1.*

**set\_env(Wx\_env::wx\_env())** -> ok

Sets the process wx environment, allows this process to use another process wx environment.

**null()** -> wxObject()

Returns the null object

**is\_null(Wx\_ref::wxObject())** -> boolean()

Returns true if object is null, false otherwise

**getObjectType(Wx\_ref::wxObject())** -> atom()

Returns the object type

**typeCast(Old::wxObject(), NewType::atom())** -> wxObject()

Casts the object to class NewType. It is needed when using functions like wxWindow:findWindow/2, which returns a generic wxObject type.

**batch(Fun::function())** -> term()

Batches all wx commands used in the fun. Improves performance of the command processing by grabbing the wxWidgets thread so that no event processing will be done before the complete batch of commands is invoked.

*See also: foldl/3, foldr/3, foreach/2, map/2.*

**foreach(Fun::function(), List::list())** -> ok

Behaves like *lists:foreach/2* but batches wx commands. See *batch/1*.

**map(Fun::function(), List::list())** -> list()

Behaves like *lists:map/2* but batches wx commands. See *batch/1*.

**foldl(Fun::function(), Acc::term(), List::list()) -> term()**

Behaves like *lists:foldl/3* but batches wx commands. See *batch/1*.

**foldr(Fun::function(), Acc::term(), List::list()) -> term()**

Behaves like *lists:foldr/3* but batches wx commands. See *batch/1*.

**create\_memory(Size::integer()) -> wx\_memory()**

Creates a memory area (of Size in bytes) which can be used by an external library (i.e. opengl). It is up to the client to keep a reference to this object so it does not get garbage collected by erlang while still in use by the external library.

This is far from erlang's intentional usage and can crash the erlang emulator. Use it carefully.

**get\_memory\_bin(Wx\_mem::wx\_memory()) -> binary()**

Returns the memory area as a binary.

**retain\_memory(Wx\_mem::wx\_memory()) -> ok**

Saves the memory from deletion until *release\_memory/1* is called. If *release\_memory/1* is not called the memory will not be garbage collected.

**release\_memory(Wx\_mem) -> term()**

**debug(Level::term()) -> ok**

Types:

**Level = none | verbose | trace | driver | [Level]**

Sets debug level. If debug level is verbose or trace each call is printed on console. If Level is driver each allocated object and deletion is printed on the console.

**demo() -> ok**

Starts a wxErlang demo if examples directory exists and is compiled

## wx\_object

Erlang module

wx\_object - Generic wx object behaviour

This is a behaviour module that can be used for "sub classing" wx objects. It works like a regular gen\_server module and creates a server per object.

NOTE: Currently no form of inheritance is implemented.

The user module should export:

init(Args) should return

{wxObject, State} | {wxObject, State, Timeout} | ignore | {stop, Reason}

handle\_call(Msg, {From, Tag}, State) should return

{reply, Reply, State} | {reply, Reply, State, Timeout} | {noreply, State} | {noreply, State, Timeout} | {stop, Reason, Reply, State}

Asynchronous window event handling:

handle\_event(#wx{}, State) should return

{noreply, State} | {noreply, State, Timeout} | {stop, Reason, State}

Info is message e.g. {'EXIT', P, R}, {nodedown, N}, ...

handle\_info(Info, State) should return , ...

{noreply, State} | {noreply, State, Timeout} | {stop, Reason, State}

When stop is returned in one of the functions above with Reason = normal | shutdown | Term, terminate(State) is called. It lets the user module clean up, it is always called when server terminates or when wxObject() in the driver is deleted. If the Parent process terminates the Module:terminate/2 function is called.

terminate(Reason, State)

Example:

```
-module(myDialog).
-export([new/2, show/1, destroy/1]). %% API
-export([init/1, handle_call/3, handle_event/2,
        handle_info/2, code_change/3, terminate/2]).
        new/2, showModal/1, destroy/1]). %% Callbacks

%% Client API
new(Parent, Msg) ->
    wx_object:start(?MODULE, [Parent,Id], []).

show(Dialog) ->
    wx_object:call(Dialog, show_modal).

destroy(Dialog) ->
    wx_object:call(Dialog, destroy).

%% Server Implementation ala gen_server
init([Parent, Str]) ->
    Dialog = wxDialog:new(Parent, 42, "Testing", []),
    ...
    wxDialog:connect(Dialog, command_button_clicked),
    {Dialog, MyState}.

handle_call(show, _From, State) ->
    wxDialog:show(State#state.win),
    {reply, ok, State};
```

```
...
handle_event(#wx{}, State) ->
    io:format("Users clicked button~n",[]),
    {noreply, State};
...
```

## Exports

**start(Mod, Args, Options) -> wxWindow()** (see module wxWindow)

Types:

**Mod** = atom()  
**Args** = term()  
**Options** = [{timeout, Timeout} | {debug, [Flag]}]  
**Flag** = trace | log | {logfile, File} | statistics | debug

Starts a generic wx\_object server and invokes Mod:init(Args) in the new process.

**start(Name, Mod, Args, Options) -> wxWindow()** (see module wxWindow)

Types:

**Name** = {local, atom()}  
**Mod** = atom()  
**Args** = term()  
**Options** = [{timeout, Timeout} | {debug, [Flag]}]  
**Flag** = trace | log | {logfile, File} | statistics | debug

Starts a generic wx\_object server and invokes Mod:init(Args) in the new process.

**start\_link(Mod, Args, Options) -> wxWindow()** (see module wxWindow)

Types:

**Mod** = atom()  
**Args** = term()  
**Options** = [{timeout, Timeout} | {debug, [Flag]}]  
**Flag** = trace | log | {logfile, File} | statistics | debug

Starts a generic wx\_object server and invokes Mod:init(Args) in the new process.

**start\_link(Name, Mod, Args, Options) -> wxWindow()** (see module wxWindow)

Types:

**Name** = {local, atom()}  
**Mod** = atom()  
**Args** = term()  
**Options** = [{timeout, Timeout} | {debug, [Flag]}]  
**Flag** = trace | log | {logfile, File} | statistics | debug

Starts a generic wx\_object server and invokes Mod:init(Args) in the new process.

```
call(Ref::wxObject() | atom() | pid(), Request::term()) -> term()
```

Make a call to a wx\_object server. The call waits until it gets a result. Invokes handle\_call(Request, From, State) in the server

```
call(Ref::wxObject() | atom() | pid(), Request::term(), Timeout::integer()) -> term()
```

Make a call to a wx\_object server with a timeout. Invokes handle\_call(Request, From, State) in server

```
cast(Ref::wxObject() | atom() | pid(), Request::term()) -> ok
```

Make a cast to a wx\_object server. Invokes handle\_cast(Request, State) in the server

```
get_pid(Ref::wxObject()) -> pid()
```

Get the pid of the object handle.

```
reply(From::tuple(), Reply::term()) -> pid()
```

Get the pid of the object handle.

## wxAcceleratorEntry

---

Erlang module

See external documentation: **wxAcceleratorEntry**.

### DATA TYPES

`wxAcceleratorEntry()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxAcceleratorEntry()`**

Equivalent to `new([])`.

**`new(X::term() | wxAcceleratorEntry())`** -> **`wxAcceleratorEntry()`**

See **external documentation**.

Alternatives:

`new([Option])` -> `wxAcceleratorEntry()`

`Option = {flags, integer()} | {keyCode, integer()} | {cmd, integer()} | {item, wxMenuItem:wxMenuItem()}`

`new(Entry::wxAcceleratorEntry())` -> `wxAcceleratorEntry()`

**`getCommand(This::wxAcceleratorEntry())`** -> **`integer()`**

See **external documentation**.

**`getFlags(This::wxAcceleratorEntry())`** -> **`integer()`**

See **external documentation**.

**`getKeyCode(This::wxAcceleratorEntry())`** -> **`integer()`**

See **external documentation**.

**`set(This::wxAcceleratorEntry(), Flags::integer(), KeyCode::integer(), Cmd::integer())`** -> **`ok`**

Equivalent to `set(This, Flags, KeyCode, Cmd, [])`.

**`set(This::wxAcceleratorEntry(), Flags::integer(), KeyCode::integer(), Cmd::integer(), Options::[Option])`** -> **`ok`**

Types:

**`Option = {item, wxMenuItem()} (see module wxMenuItem)`**

See **external documentation**.

**destroy(This::wxAcceleratorEntry()) -> ok**

Destroys this object, do not use object again

## wxAcceleratorTable

---

Erlang module

See external documentation: **wxAcceleratorTable**.

### DATA TYPES

`wxAcceleratorTable()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxAcceleratorTable()`

See external documentation.

`new(N::integer(), Entries::[wxAcceleratorEntry() (see module wxAcceleratorEntry)]) -> wxAcceleratorTable()`

See external documentation.

`ok(This::wxAcceleratorTable()) -> bool()`

See external documentation.

`destroy(This::wxAcceleratorTable()) -> ok`

Destroys this object, do not use object again



## wxArtProvider

---

Erlang module

See external documentation: **wxArtProvider**.

### DATA TYPES

`wxArtProvider()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`getBitmap(Id::string()) -> wxBitmap()` (see module `wxBitmap`)

Equivalent to `getBitmap(Id, [])`.

`getBitmap(Id::string(), Options::[Option]) -> wxBitmap()` (see module `wxBitmap`)

Types:

`Option = {client, string()} | {size, {W::integer(), H::integer()}}`

See external documentation.

`getIcon(Id::string()) -> wxIcon()` (see module `wxIcon`)

Equivalent to `getIcon(Id, [])`.

`getIcon(Id::string(), Options::[Option]) -> wxIcon()` (see module `wxIcon`)

Types:

`Option = {client, string()} | {size, {W::integer(), H::integer()}}`

See external documentation.

## wxAuiDockArt

---

Erlang module

See external documentation: **wxAuiDockArt**.

### DATA TYPES

`wxAuiDockArt ( )`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

# wxGuiManager

Erlang module

See external documentation: **wxGuiManager**.

This class is derived (and can use functions) from:  
*wxEvtHandler*

## DATA TYPES

`wxGuiManager()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**`new()`** -> **`wxGuiManager()`**

Equivalent to *new([])*.

**`new(Options::[Option])`** -> **`wxGuiManager()`**

Types:

**Option** = {managed\_wnd, wxWindow() (see module wxWindow)} | {flags, integer()}

See external documentation.

**`addPane(This::wxGuiManager(), Window::wxWindow() (see module wxWindow))`** -> **`bool()`**

Equivalent to *addPane(This, Window, [])*.

**`addPane(This::wxGuiManager(), Window::wxWindow() (see module wxWindow), X::term())`** -> **`bool()`**

See external documentation.

Alternatives:

**`addPane(This::wxGuiManager(), Window::wxWindow:wxWindow(), [Option])`** -> **`bool()`**  
Option = {direction, integer()} | {caption, string() }

**`addPane(This::wxGuiManager(), Window::wxWindow:wxWindow(), Pane_info::wxGuiPaneInfo:wxGuiPaneInfo())`** -> **`bool()`**

**`addPane(This::wxGuiManager(), Window::wxWindow() (see module wxWindow), Pane_info::wxGuiPaneInfo() (see module wxGuiPaneInfo), Drop_pos:: {X::integer(), Y::integer()})`** -> **`bool()`**

See external documentation.

**`detachPane(This::wxGuiManager(), Window::wxWindow() (see module wxWindow))`** -> **`bool()`**

See external documentation.

`getAllPanels(This::wxAuiManager()) -> wxAuiPaneInfoArray()` (see module `wxAuiPaneInfoArray`)

See external documentation.

`getArtProvider(This::wxAuiManager()) -> wxAuiDockArt()` (see module `wxAuiDockArt`)

See external documentation.

`getDockSizeConstraint(This::wxAuiManager()) -> {Width_pct::float(), Height_pct::float()}`

See external documentation.

`getFlags(This::wxAuiManager()) -> integer()`

See external documentation.

`getManagedWindow(This::wxAuiManager()) -> wxWindow()` (see module `wxWindow`)

See external documentation.

`getManager(Window::wxWindow() (see module wxWindow)) -> wxAuiManager()`

See external documentation.

`getPane(This::wxAuiManager(), X::string() | term()) -> wxAuiPaneInfo()` (see module `wxAuiPaneInfo`)

See external documentation.

Alternatives:

`getPane(This::wxAuiManager(), Name::string()) -> wxAuiPaneInfo:wxAuiPaneInfo()`

`getPane(This::wxAuiManager(), Window::wxWindow:wxWindow()) -> wxAuiPaneInfo:wxAuiPaneInfo()`

`hideHint(This::wxAuiManager()) -> ok`

See external documentation.

`insertPane(This::wxAuiManager(), Window::wxWindow() (see module wxWindow), Insert_location::wxAuiPaneInfo() (see module wxAuiPaneInfo)) -> bool()`

Equivalent to `insertPane(This, Window, Insert_location, [])`.

`insertPane(This::wxAuiManager(), Window::wxWindow() (see module wxWindow), Insert_location::wxAuiPaneInfo() (see module wxAuiPaneInfo), Options::[Option]) -> bool()`

Types:

`Option = {insert_level, integer()}`

See external documentation.

`loadPaneInfo(This::wxAuiManager(), Pane_part::string(), Pane::wxAuiPaneInfo() (see module wxAuiPaneInfo)) -> ok`

See external documentation.

`loadPerspective(This::wxAuiManager(), Perspective::string()) -> bool()`

Equivalent to `loadPerspective(This, Perspective, [])`.

`loadPerspective(This::wxAuiManager(), Perspective::string(), Options::[Option]) -> bool()`

Types:

`Option = {update, bool()}`

See external documentation.

`savePaneInfo(This::wxAuiManager(), Pane::wxAuiPaneInfo() (see module wxAuiPaneInfo)) -> string()`

See external documentation.

`savePerspective(This::wxAuiManager()) -> string()`

See external documentation.

`setArtProvider(This::wxAuiManager(), Art_provider::wxAuiDockArt() (see module wxAuiDockArt)) -> ok`

See external documentation.

`setDockSizeConstraint(This::wxAuiManager(), Width_pct::float(), Height_pct::float()) -> ok`

See external documentation.

`setFlags(This::wxAuiManager(), Flags::integer()) -> ok`

See external documentation.

`setManagedWindow(This::wxAuiManager(), Managed_wnd::wxWindow() (see module wxWindow)) -> ok`

See external documentation.

`showHint(This::wxAuiManager(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> ok`

See external documentation.

`unInit(This::wxAuiManager()) -> ok`

See external documentation.

`update(This::wxAuiManager()) -> ok`

See external documentation.

## **wxAuiManager**

---

**destroy(This::wxAuiManager()) -> ok**

Destroys this object, do not use object again

# wxAuiManagerEvent

Erlang module

See external documentation: **wxAuiManagerEvent**.

Use *wxEvtHandler::connect/3* with EventType:

*au\_i\_pane\_button, au\_i\_pane\_close, au\_i\_pane\_maximize, au\_i\_pane\_restore, au\_i\_render, au\_i\_find\_manager*

See also the message variant *#wxAuiManager{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

`wxAuiManagerEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**setManager(This::wxAuiManagerEvent(), Mgr::wxAuiManager())** (see module **wxAuiManager**) -> ok

See external documentation.

**getManager(This::wxAuiManagerEvent())** -> wxAuiManager() (see module **wxAuiManager**)

See external documentation.

**setPane(This::wxAuiManagerEvent(), P::wxAuiPaneInfo())** (see module **wxAuiPaneInfo**) -> ok

See external documentation.

**getPane(This::wxAuiManagerEvent())** -> wxAuiPaneInfo() (see module **wxAuiPaneInfo**)

See external documentation.

**setButton(This::wxAuiManagerEvent(), B::integer())** -> ok

See external documentation.

**getButton(This::wxAuiManagerEvent())** -> integer()

See external documentation.

**setDC(This::wxAuiManagerEvent(), Pdc::wxDC())** (see module **wxDC**) -> ok

See external documentation.

**getDC(This::wxAuiManagerEvent()) -> wxDC()** (see module wxDC)

See [external documentation](#).

**veto(This::wxAuiManagerEvent()) -> ok**

Equivalent to *veto(This, [])*.

**veto(This::wxAuiManagerEvent(), Options::[Option]) -> ok**

Types:

**Option = {veto, bool()}**

See [external documentation](#).

**getVeto(This::wxAuiManagerEvent()) -> bool()**

See [external documentation](#).

**setCanVeto(This::wxAuiManagerEvent(), Can\_veto::bool()) -> ok**

See [external documentation](#).

**canVeto(This::wxAuiManagerEvent()) -> bool()**

See [external documentation](#).



# wxAuiNotebook

Erlang module

See external documentation: **wxAuiNotebook**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxAuiNotebook()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxAuiNotebook()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxAuiNotebook()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxAuiNotebook()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`addPage(This::wxAuiNotebook(), Page::wxWindow() (see module wxWindow), Caption::string()) -> bool()`

Equivalent to `addPage(This, Page, Caption, [])`.

`addPage(This::wxAuiNotebook(), Page::wxWindow() (see module wxWindow), Caption::string(), Options::[Option]) -> bool()`

Types:

**Option** = {select, bool()} | {bitmap, wxBitmap() (see module wxBitmap)}

See external documentation.

`create(This::wxAuiNotebook(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

```
create(This::wxAuiNotebook(), Parent::wxWindow() (see module wxWindow),
Options::[Option]) -> bool()
```

Types:

```
Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style,
integer()}
```

See external documentation.

```
deletePage(This::wxAuiNotebook(), Page::integer()) -> bool()
```

See external documentation.

```
getArtProvider(This::wxAuiNotebook()) -> wxAuiTabArt() (see module
wxAuiTabArt)
```

See external documentation.

```
getPage(This::wxAuiNotebook(), Page_idx::integer()) -> wxWindow() (see module
wxWindow)
```

See external documentation.

```
getPageBitmap(This::wxAuiNotebook(), Page_idx::integer()) -> wxBitmap() (see
module wxBitmap)
```

See external documentation.

```
getPageCount(This::wxAuiNotebook()) -> integer()
```

See external documentation.

```
getPageIndex(This::wxAuiNotebook(), Page_wnd::wxWindow() (see module
wxWindow)) -> integer()
```

See external documentation.

```
getPageText(This::wxAuiNotebook(), Page_idx::integer()) -> string()
```

See external documentation.

```
getSelection(This::wxAuiNotebook()) -> integer()
```

See external documentation.

```
insertPage(This::wxAuiNotebook(), Page_idx::integer(), Page::wxWindow() (see
module wxWindow), Caption::string()) -> bool()
```

Equivalent to *insertPage(This, Page\_idx, Page, Caption, [])*.

```
insertPage(This::wxAuiNotebook(), Page_idx::integer(), Page::wxWindow() (see
module wxWindow), Caption::string(), Options::[Option]) -> bool()
```

Types:

```
Option = {select, bool()} | {bitmap, wxBitmap() (see module wxBitmap)}
```

See external documentation.

**removePage**(This::wxAuiNotebook(), Page::integer()) -> bool()

See external documentation.

**setArtProvider**(This::wxAuiNotebook(), Art::wxAuiTabArt() (see module wxAuiTabArt)) -> ok

See external documentation.

**setFont**(This::wxAuiNotebook(), Font::wxFont() (see module wxFont)) -> bool()

See external documentation.

**setPageBitmap**(This::wxAuiNotebook(), Page::integer(), Bitmap::wxBitmap() (see module wxBitmap)) -> bool()

See external documentation.

**setPageText**(This::wxAuiNotebook(), Page::integer(), Text::string()) -> bool()

See external documentation.

**setSelection**(This::wxAuiNotebook(), New\_page::integer()) -> integer()

See external documentation.

**setTabCtrlHeight**(This::wxAuiNotebook(), Height::integer()) -> ok

See external documentation.

**setUniformBitmapSize**(This::wxAuiNotebook(), Size::{W::integer(), H::integer()}) -> ok

See external documentation.

**destroy**(This::wxAuiNotebook()) -> ok

Destroys this object, do not use object again

## wxAuiNotebookEvent

---

Erlang module

See external documentation: **wxAuiNotebookEvent**.

Use *wxEvtHandler:connect/3* with EventType:

<i>command_auiNotebook_page_close,</i>	<i>command_auiNotebook_page_changed,</i>
<i>command_auiNotebook_page_changing,</i>	<i>command_auiNotebook_button,</i>
<i>command_auiNotebook_end_drag,</i>	<i>command_auiNotebook_begin_drag,</i>
<i>command_auiNotebook_drag_motion,</i>	<i>command_auiNotebook_allow_dnd,</i>
<i>command_auiNotebook_tab_middle_down,</i>	<i>command_auiNotebook_tab_middle_up,</i>
<i>command_auiNotebook_tab_right_down,</i>	<i>command_auiNotebook_tab_right_up,</i>
<i>command_auiNotebook_page_closed,</i>	<i>command_auiNotebook_drag_done,</i>
<i>command_auiNotebook_bg_dclick</i>	

See also the message variant *#wxAuiNotebook{}* event record type.

This class is derived (and can use functions) from:

*wxNotifyEvent*

*wxCommandEvent*

*wxEvent*

## DATA TYPES

*wxAuiNotebookEvent()*

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**setSelection(This::wxAuiNotebookEvent(), S::integer()) -> ok**

See external documentation.

**getSelection(This::wxAuiNotebookEvent()) -> integer()**

See external documentation.

**setOldSelection(This::wxAuiNotebookEvent(), S::integer()) -> ok**

See external documentation.

**getOldSelection(This::wxAuiNotebookEvent()) -> integer()**

See external documentation.

**setDragSource(This::wxAuiNotebookEvent(), S::wxAuiNotebook() (see module wxAuiNotebook)) -> ok**

See external documentation.

**getDragSource(This::wxAuiNotebookEvent()) -> wxAuiNotebook() (see module wxAuiNotebook)**

See external documentation.

---

# wxAuiPanelInfo

---

Erlang module

See external documentation: **wxAuiPanelInfo**.

## DATA TYPES

`wxAuiPanelInfo()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**`new()`** -> **`wxAuiPanelInfo()`**

See external documentation.

**`new(C::wxAuiPanelInfo())`** -> **`wxAuiPanelInfo()`**

See external documentation.

**`bestSize(This::wxAuiPanelInfo(), Size::{W::integer(), H::integer()})`** -> **`wxAuiPanelInfo()`**

See external documentation.

**`bestSize(This::wxAuiPanelInfo(), X::integer(), Y::integer())`** -> **`wxAuiPanelInfo()`**

See external documentation.

**`bottom(This::wxAuiPanelInfo())`** -> **`wxAuiPanelInfo()`**

See external documentation.

**`bottomDockable(This::wxAuiPanelInfo())`** -> **`wxAuiPanelInfo()`**

Equivalent to *`bottomDockable(This, [])`*.

**`bottomDockable(This::wxAuiPanelInfo(), Options::[Option])`** -> **`wxAuiPanelInfo()`**

Types:

**`Option = {b, bool()}`**

See external documentation.

**`caption(This::wxAuiPanelInfo(), C::string())`** -> **`wxAuiPanelInfo()`**

See external documentation.

**`captionVisible(This::wxAuiPanelInfo())`** -> **`wxAuiPanelInfo()`**

Equivalent to *`captionVisible(This, [])`*.

`captionVisible(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

**Option** = {visible, bool()}

See external documentation.

`centre(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

See external documentation.

`centrePane(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

See external documentation.

`closeButton(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

Equivalent to `closeButton(This, [])`.

`closeButton(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

**Option** = {visible, bool()}

See external documentation.

`defaultPane(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

See external documentation.

`destroyOnClose(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

Equivalent to `destroyOnClose(This, [])`.

`destroyOnClose(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

**Option** = {b, bool()}

See external documentation.

`direction(This::wxAuiPaneInfo(), Direction::integer()) -> wxAuiPaneInfo()`

See external documentation.

`dock(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

See external documentation.

`dockable(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

Equivalent to `dockable(This, [])`.

`dockable(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

**Option** = {b, bool()}

See [external documentation](#).

```
fixed(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()
```

See [external documentation](#).

```
float(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()
```

See [external documentation](#).

```
floatable(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()
```

Equivalent to *floatable(This, [])*.

```
floatable(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()
```

Types:

**Option** = {b, bool()}

See [external documentation](#).

```
floatingPosition(This::wxAuiPaneInfo(), Pos::{X::integer(), Y::integer()}) ->  
wxAuiPaneInfo()
```

See [external documentation](#).

```
floatingPosition(This::wxAuiPaneInfo(), X::integer(), Y::integer()) ->  
wxAuiPaneInfo()
```

See [external documentation](#).

```
floatingSize(This::wxAuiPaneInfo(), Size::{W::integer(), H::integer()}) ->  
wxAuiPaneInfo()
```

See [external documentation](#).

```
floatingSize(This::wxAuiPaneInfo(), X::integer(), Y::integer()) ->  
wxAuiPaneInfo()
```

See [external documentation](#).

```
gripper(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()
```

Equivalent to *gripper(This, [])*.

```
gripper(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()
```

Types:

**Option** = {visible, bool()}

See [external documentation](#).

```
gripperTop(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()
```

Equivalent to *gripperTop(This, [])*.

`gripperTop(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

`Option = {attop, bool()}`

See external documentation.

`hasBorder(This::wxAuiPaneInfo()) -> bool()`

See external documentation.

`hasCaption(This::wxAuiPaneInfo()) -> bool()`

See external documentation.

`hasCloseButton(This::wxAuiPaneInfo()) -> bool()`

See external documentation.

`hasFlag(This::wxAuiPaneInfo(), Flag::integer()) -> bool()`

See external documentation.

`hasGripper(This::wxAuiPaneInfo()) -> bool()`

See external documentation.

`hasGripperTop(This::wxAuiPaneInfo()) -> bool()`

See external documentation.

`hasMaximizeButton(This::wxAuiPaneInfo()) -> bool()`

See external documentation.

`hasMinimizeButton(This::wxAuiPaneInfo()) -> bool()`

See external documentation.

`hasPinButton(This::wxAuiPaneInfo()) -> bool()`

See external documentation.

`hide(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

See external documentation.

`isBottomDockable(This::wxAuiPaneInfo()) -> bool()`

See external documentation.

`isDocked(This::wxAuiPaneInfo()) -> bool()`

See external documentation.



**isFixed**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isFloatable**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isFloating**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isLeftDockable**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isMovable**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isOk**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isResizable**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isRightDockable**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isShown**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isToolbar**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**isTopDockable**(This::wxAuiPaneInfo()) -> bool()

See external documentation.

**layer**(This::wxAuiPaneInfo(), Layer::integer()) -> wxAuiPaneInfo()

See external documentation.

**left**(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()

See external documentation.

**leftDockable**(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()

Equivalent to *leftDockable(This, [])*.

`leftDockable(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

`Option = {b, bool()}`

See external documentation.

`maxSize(This::wxAuiPaneInfo(), Size::{W::integer(), H::integer()}) -> wxAuiPaneInfo()`

See external documentation.

`maxSize(This::wxAuiPaneInfo(), X::integer(), Y::integer()) -> wxAuiPaneInfo()`

See external documentation.

`maximizeButton(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

Equivalent to `maximizeButton(This, [])`.

`maximizeButton(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

`Option = {visible, bool()}`

See external documentation.

`minSize(This::wxAuiPaneInfo(), Size::{W::integer(), H::integer()}) -> wxAuiPaneInfo()`

See external documentation.

`minSize(This::wxAuiPaneInfo(), X::integer(), Y::integer()) -> wxAuiPaneInfo()`

See external documentation.

`minimizeButton(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

Equivalent to `minimizeButton(This, [])`.

`minimizeButton(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

`Option = {visible, bool()}`

See external documentation.

`movable(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

Equivalent to `movable(This, [])`.

`movable(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

`Option = {b, bool()}`

See external documentation.

**name**(This::wxAuiPaneInfo(), N::string()) -> wxAuiPaneInfo()

See external documentation.

**paneBorder**(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()

Equivalent to *paneBorder*(This, []).

**paneBorder**(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()

Types:

**Option** = {visible, bool()}

See external documentation.

**pinButton**(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()

Equivalent to *pinButton*(This, []).

**pinButton**(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()

Types:

**Option** = {visible, bool()}

See external documentation.

**position**(This::wxAuiPaneInfo(), Pos::integer()) -> wxAuiPaneInfo()

See external documentation.

**resizable**(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()

Equivalent to *resizable*(This, []).

**resizable**(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()

Types:

**Option** = {resizable, bool()}

See external documentation.

**right**(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()

See external documentation.

**rightDockable**(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()

Equivalent to *rightDockable*(This, []).

**rightDockable**(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()

Types:

**Option** = {b, bool()}

See external documentation.

`row(This::wxAuiPaneInfo(), Row::integer()) -> wxAuiPaneInfo()`

See external documentation.

`safeSet(This::wxAuiPaneInfo(), Source::wxAuiPaneInfo()) -> ok`

See external documentation.

`setFlag(This::wxAuiPaneInfo(), Flag::integer(), Option_state::bool()) -> wxAuiPaneInfo()`

See external documentation.

`show(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

Equivalent to `show(This, [])`.

`show(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

**Option** = {show, bool()}

See external documentation.

`toolbarPane(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

See external documentation.

`top(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

See external documentation.

`topDockable(This::wxAuiPaneInfo()) -> wxAuiPaneInfo()`

Equivalent to `topDockable(This, [])`.

`topDockable(This::wxAuiPaneInfo(), Options::[Option]) -> wxAuiPaneInfo()`

Types:

**Option** = {b, bool()}

See external documentation.

`window(This::wxAuiPaneInfo(), W::wxWindow() (see module wxWindow)) -> wxAuiPaneInfo()`

See external documentation.

`destroy(This::wxAuiPaneInfo()) -> ok`

Destroys this object, do not use object again

## wxAuiTabArt

---

Erlang module

See external documentation: **wxAuiTabArt**.

### DATA TYPES

`wxAuiTabArt()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxBitmap

---

Erlang module

See external documentation: **wxBitmap**.

### DATA TYPES

`wxBitmap()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()` -> `wxBitmap()`**

See external documentation.

**`new(X::string() | term())` -> `wxBitmap()`**

See external documentation.

Alternatives:

`new(Filename::string())` -> `new(Filename, [])`

`new(Image::wxImage:wxImage())` -> `new(Image, [])`

**`new(X::integer() | string() | term(), X::integer() | term())` -> `wxBitmap()`**

See external documentation.

Alternatives:

`new(Width::integer(), Height::integer())` -> `new(Width,Height, [])`

`new(Filename::string(), [Option])` -> `wxBitmap()`

Option = {type, WxBitmapType}

WxBitmapType = integer()

WxBitmapType is one of ?wxBITMAP\_TYPE\_INVALID | ?wxBITMAP\_TYPE\_BMP | ?wxBITMAP\_TYPE\_BMP\_RESOURCE | ?wxBITMAP\_TYPE\_RESOURCE | ?wxBITMAP\_TYPE\_ICO | ?wxBITMAP\_TYPE\_ICO\_RESOURCE | ?wxBITMAP\_TYPE\_CUR | ?wxBITMAP\_TYPE\_CUR\_RESOURCE | ?wxBITMAP\_TYPE\_XBM | ?wxBITMAP\_TYPE\_XBM\_DATA | ?wxBITMAP\_TYPE\_XPM | ?wxBITMAP\_TYPE\_XPM\_DATA | ?wxBITMAP\_TYPE\_TIF | ?wxBITMAP\_TYPE\_TIF\_RESOURCE | ?wxBITMAP\_TYPE\_GIF | ?wxBITMAP\_TYPE\_GIF\_RESOURCE | ?wxBITMAP\_TYPE\_PNG | ?wxBITMAP\_TYPE\_PNG\_RESOURCE | ?wxBITMAP\_TYPE\_JPEG | ?wxBITMAP\_TYPE\_JPEG\_RESOURCE | ?wxBITMAP\_TYPE\_PNM | ?wxBITMAP\_TYPE\_PNM\_RESOURCE | ?wxBITMAP\_TYPE\_PCX | ?wxBITMAP\_TYPE\_PCX\_RESOURCE | ?wxBITMAP\_TYPE\_PICT | ?wxBITMAP\_TYPE\_PICT\_RESOURCE | ?wxBITMAP\_TYPE\_ICON | ?wxBITMAP\_TYPE\_ICON\_RESOURCE | ?wxBITMAP\_TYPE\_ANI | ?wxBITMAP\_TYPE\_IFF | ?wxBITMAP\_TYPE\_TGA | ?wxBITMAP\_TYPE\_MACCURSOR | ?wxBITMAP\_TYPE\_MACCURSOR\_RESOURCE | ?wxBITMAP\_TYPE\_ANY

`new(Image::wxImage:wxImage(), [Option])` -> `wxBitmap()`

Option = {depth, integer()}

```
new(X::binary() | integer(), X::integer(), X::integer() | term()) ->
wxBitmap()
```

See external documentation.

Alternatives:

```
new(Bits::binary(), Width::integer(), Height::integer()) ->
new(Bits,Width,Height, [])
```

```
new(Width::integer(), Height::integer(), [Option]) -> wxBitmap()
Option = {depth, integer()}
```

```
new(Bits::binary(), Width::integer(), Height::integer(), Options::[Option]) -
> wxBitmap()
```

Types:

**Option** = {depth, integer()}

See external documentation.

```
convertToImage(This::wxBitmap()) -> wxImage() (see module wxImage)
```

See external documentation.

```
copyFromIcon(This::wxBitmap(), Icon::wxIcon() (see module wxIcon)) -> bool()
```

See external documentation.

```
create(This::wxBitmap(), Width::integer(), Height::integer()) -> bool()
```

Equivalent to *create(This, Width, Height, [])*.

```
create(This::wxBitmap(), Width::integer(), Height::integer(), Options::
[Option]) -> bool()
```

Types:

**Option** = {depth, integer()}

See external documentation.

```
getDepth(This::wxBitmap()) -> integer()
```

See external documentation.

```
getHeight(This::wxBitmap()) -> integer()
```

See external documentation.

```
getPalette(This::wxBitmap()) -> wxPalette() (see module wxPalette)
```

See external documentation.

```
getMask(This::wxBitmap()) -> wxMask() (see module wxMask)
```

See external documentation.

**getWidth(***This::wxBitmap()***) -> integer()**

See [external documentation](#).

**getSubBitmap(***This::wxBitmap()***, Rect::***{X::integer(), Y::integer(), W::integer(), H::integer()}***) -> wxBitmap()**

See [external documentation](#).

**loadFile(***This::wxBitmap()***, Name::string()****) -> bool()**

Equivalent to *loadFile(This, Name, [])*.

**loadFile(***This::wxBitmap()***, Name::string(), Options::***[Option]***) -> bool()**

Types:

**Option** = {type, WxBitmapType}

**WxBitmapType** = integer()

See [external documentation](#).

WxBitmapType is one of ?wxBITMAP\_TYPE\_INVALID | ?wxBITMAP\_TYPE\_BMP | ?wxBITMAP\_TYPE\_BMP\_RESOURCE | ?wxBITMAP\_TYPE\_RESOURCE | ?wxBITMAP\_TYPE\_ICO | ?wxBITMAP\_TYPE\_ICO\_RESOURCE | ?wxBITMAP\_TYPE\_CUR | ?wxBITMAP\_TYPE\_CUR\_RESOURCE | ?wxBITMAP\_TYPE\_XBM | ?wxBITMAP\_TYPE\_XBM\_DATA | ?wxBITMAP\_TYPE\_XPM | ?wxBITMAP\_TYPE\_XPM\_DATA | ?wxBITMAP\_TYPE\_TIF | ?wxBITMAP\_TYPE\_TIF\_RESOURCE | ?wxBITMAP\_TYPE\_GIF | ?wxBITMAP\_TYPE\_GIF\_RESOURCE | ?wxBITMAP\_TYPE\_PNG | ?wxBITMAP\_TYPE\_PNG\_RESOURCE | ?wxBITMAP\_TYPE\_JPEG | ?wxBITMAP\_TYPE\_JPEG\_RESOURCE | ?wxBITMAP\_TYPE\_PNM | ?wxBITMAP\_TYPE\_PNM\_RESOURCE | ?wxBITMAP\_TYPE\_PCX | ?wxBITMAP\_TYPE\_PCX\_RESOURCE | ?wxBITMAP\_TYPE\_PICT | ?wxBITMAP\_TYPE\_PICT\_RESOURCE | ?wxBITMAP\_TYPE\_ICON | ?wxBITMAP\_TYPE\_ICON\_RESOURCE | ?wxBITMAP\_TYPE\_ANI | ?wxBITMAP\_TYPE\_IFF | ?wxBITMAP\_TYPE\_TGA | ?wxBITMAP\_TYPE\_MACCURLSOR | ?wxBITMAP\_TYPE\_MACCURLSOR\_RESOURCE | ?wxBITMAP\_TYPE\_ANY

**ok(***This::wxBitmap()***) -> bool()**

See [external documentation](#).

**saveFile(***This::wxBitmap()***, Name::string(), Type::WxBitmapType)** **-> bool()**

Equivalent to *saveFile(This, Name, Type, [])*.

**saveFile(***This::wxBitmap()***, Name::string(), Type::WxBitmapType, Options::***[Option]***) -> bool()**

Types:

**Option** = {palette, wxPalette() (see module wxPalette)}

**WxBitmapType** = integer()

See [external documentation](#).

WxBitmapType is one of ?wxBITMAP\_TYPE\_INVALID | ?wxBITMAP\_TYPE\_BMP | ?wxBITMAP\_TYPE\_BMP\_RESOURCE | ?wxBITMAP\_TYPE\_RESOURCE | ?wxBITMAP\_TYPE\_ICO | ?wxBITMAP\_TYPE\_ICO\_RESOURCE | ?wxBITMAP\_TYPE\_CUR | ?wxBITMAP\_TYPE\_CUR\_RESOURCE | ?wxBITMAP\_TYPE\_XBM | ?wxBITMAP\_TYPE\_XBM\_DATA | ?wxBITMAP\_TYPE\_XPM | ?wxBITMAP\_TYPE\_XPM\_DATA | ?wxBITMAP\_TYPE\_TIF | ?wxBITMAP\_TYPE\_TIF\_RESOURCE | ?wxBITMAP\_TYPE\_GIF | ?wxBITMAP\_TYPE\_GIF\_RESOURCE | ?wxBITMAP\_TYPE\_PNG | ?



wxBITMAP\_TYPE\_PNG\_RESOURCE | ?wxBITMAP\_TYPE\_JPEG | ?wxBITMAP\_TYPE\_JPEG\_RESOURCE  
| ?wxBITMAP\_TYPE\_PNM | ?wxBITMAP\_TYPE\_PNM\_RESOURCE | ?wxBITMAP\_TYPE\_PCX | ?  
wxBITMAP\_TYPE\_PCX\_RESOURCE | ?wxBITMAP\_TYPE\_PICT | ?wxBITMAP\_TYPE\_PICT\_RESOURCE  
| ?wxBITMAP\_TYPE\_ICON | ?wxBITMAP\_TYPE\_ICON\_RESOURCE | ?wxBITMAP\_TYPE\_ANI  
| ?wxBITMAP\_TYPE\_IFF | ?wxBITMAP\_TYPE\_TGA | ?wxBITMAP\_TYPE\_MACCURSOR | ?  
wxBITMAP\_TYPE\_MACCURSOR\_RESOURCE | ?wxBITMAP\_TYPE\_ANY

**setDepth(This::wxBitmap(), Depth::integer()) -> ok**

See external documentation.

**setHeight(This::wxBitmap(), Height::integer()) -> ok**

See external documentation.

**setMask(This::wxBitmap(), Mask::wxMask() (see module wxMask)) -> ok**

See external documentation.

**setPalette(This::wxBitmap(), Palette::wxPalette() (see module wxPalette)) ->  
ok**

See external documentation.

**setWidth(This::wxBitmap(), Width::integer()) -> ok**

See external documentation.

**destroy(This::wxBitmap()) -> ok**

Destroys this object, do not use object again

## wxBitmapButton

---

Erlang module

See external documentation: **wxBitmapButton**.

This class is derived (and can use functions) from:

*wxButton*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxBitmapButton()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxBitmapButton()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(),  
Bitmap::wxBitmap() (see module wxBitmap)) -> wxBitmapButton()`

Equivalent to `new(Parent, Id, Bitmap, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(),  
Bitmap::wxBitmap() (see module wxBitmap), Options::[Option]) ->  
wxBitmapButton()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} |  
{validator, wx() (see module wx)}

See external documentation.

`create(This::wxBitmapButton(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Bitmap::wxBitmap() (see module wxBitmap)) -> bool()`

Equivalent to `create(This, Parent, Id, Bitmap, [])`.

`create(This::wxBitmapButton(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Bitmap::wxBitmap() (see module wxBitmap), Options::[Option]) -  
> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} |  
{validator, wx() (see module wx)}

See external documentation.

`getBitmapDisabled(This::wxBitmapButton()) -> wxBitmap()` (see module `wxBitmap`)

See external documentation.

`getBitmapFocus(This::wxBitmapButton()) -> wxBitmap()` (see module `wxBitmap`)

See external documentation.

`getBitmapLabel(This::wxBitmapButton()) -> wxBitmap()` (see module `wxBitmap`)

See external documentation.

`getBitmapSelected(This::wxBitmapButton()) -> wxBitmap()` (see module `wxBitmap`)

See external documentation.

`setBitmapDisabled(This::wxBitmapButton(), Disabled::wxBitmap()` (see module `wxBitmap`)) `-> ok`

See external documentation.

`setBitmapFocus(This::wxBitmapButton(), Focus::wxBitmap()` (see module `wxBitmap`)) `-> ok`

See external documentation.

`setBitmapLabel(This::wxBitmapButton(), Bitmap::wxBitmap()` (see module `wxBitmap`)) `-> ok`

See external documentation.

`setBitmapSelected(This::wxBitmapButton(), Sel::wxBitmap()` (see module `wxBitmap`)) `-> ok`

See external documentation.

`destroy(This::wxBitmapButton()) -> ok`

Destroys this object, do not use object again

## wxBitmapDataObject

---

Erlang module

See external documentation: **wxBitmapDataObject**.

This class is derived (and can use functions) from:  
*wxDataObject*

### DATA TYPES

`wxBitmapDataObject()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()` -> `wxBitmapDataObject()`**

Equivalent to `new([])`.

**`new(X::term())` -> `wxBitmapDataObject()`**

See **external documentation**.

Alternatives:

`new([Option])` -> `wxBitmapDataObject()`

Option = {bitmap, wxBitmap:wxBitmap() }

`new(Bitmap::wxBitmap:wxBitmap())` -> `wxBitmapDataObject()`

**`getBitmap(This::wxBitmapDataObject())` -> `wxBitmap()` (see module `wxBitmap`)**

See **external documentation**.

**`setBitmap(This::wxBitmapDataObject(), Bitmap::wxBitmap())` (see module `wxBitmap`) -> `ok`**

See **external documentation**.

**`destroy(This::wxBitmapDataObject())` -> `ok`**

Destroys this object, do not use object again

## wxBoxSizer

---

Erlang module

See external documentation: **wxBoxSizer**.

This class is derived (and can use functions) from:  
*wxSizer*

### DATA TYPES

`wxBoxSizer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Orient::integer()) -> wxBoxSizer()`

See external documentation.

`getOrientation(This::wxBoxSizer()) -> integer()`

See external documentation.

`destroy(This::wxBoxSizer()) -> ok`

Destroys this object, do not use object again

## wxBush

---

Erlang module

See external documentation: **wxBush**.

### DATA TYPES

`wxBush()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new()` -> `wxBush()`

See external documentation.

`new(X::term())` -> `wxBush()`

See external documentation.

Alternatives:

`new(Colour::wx:colour())` -> `new(Colour, [])`

`new(StippleBitmap::wxBitmap:wxBitmap())` -> `wxBush()`

`new(Colour::colour() (see module wx), Options::[Option])` -> `wxBush()`

Types:

**Option** = {style, integer()}

See external documentation.

`getColour(This::wxBush())` -> `colour() (see module wx)`

See external documentation.

`getStipple(This::wxBush())` -> `wxBitmap() (see module wxBitmap)`

See external documentation.

`getStyle(This::wxBush())` -> `integer()`

See external documentation.

`isHatch(This::wxBush())` -> `bool()`

See external documentation.

`isOk(This::wxBush())` -> `bool()`

See external documentation.

**setColour(This::wxBrush(), Col::colour() (see module wx)) -> ok**

See external documentation.

**setColour(This::wxBrush(), R::integer(), G::integer(), B::integer()) -> ok**

See external documentation.

**setStipple(This::wxBrush(), Stipple::wxBitmap() (see module wxBitmap)) -> ok**

See external documentation.

**setStyle(This::wxBrush(), Style::integer()) -> ok**

See external documentation.

**destroy(This::wxBrush()) -> ok**

Destroys this object, do not use object again

## wxBufferedDC

---

Erlang module

See external documentation: **wxBufferedDC**.

This class is derived (and can use functions) from:

*wxMemoryDC*

*wxDC*

## DATA TYPES

`wxBufferedDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**`new()`** -> **`wxBufferedDC()`**

See **external documentation**.

**`new(Dc::wxDC())`** (see module **`wxDC`**) -> **`wxBufferedDC()`**

Equivalent to `new(Dc, [])`.

**`new(Dc::wxDC() (see module wxDC), X::term())`** -> **`wxBufferedDC()`**

See **external documentation**.

Alternatives:

`new(Dc::wxDC:wxDC(), Area::{W::integer(),H::integer()})` -> `new(Dc,Area, [])`

`new(Dc::wxDC:wxDC(), [Option])` -> `wxBufferedDC()`

Option = {buffer, wxBitmap:wxBitmap()} | {style, integer()}

**`new(Dc::wxDC() (see module wxDC), Area::{W::integer(), H::integer()}, Options::[Option])`** -> **`wxBufferedDC()`**

Types:

**Option** = {style, integer()}

See **external documentation**.

**`init(This::wxBufferedDC(), Dc::wxDC() (see module wxDC))`** -> **`ok`**

Equivalent to `init(This, Dc, [])`.

**`init(This::wxBufferedDC(), Dc::wxDC() (see module wxDC), X::term())`** -> **`ok`**

See **external documentation**.

Alternatives:

`init(This::wxBufferedDC(), Dc::wxDC:wxDC(), Area::{W::integer(),H::integer()})` -> `init(This,Dc,Area, [])`



```
init(This::wxBufferedDC(), Dc::wxDC:wxDC(), [Option]) -> ok  
Option = {buffer, wxBitmap:wxBitmap()} | {style, integer()}
```

```
init(This::wxBufferedDC(), Dc::wxDC() (see module wxDC), Area::{W::integer(),  
H::integer()}, Options::[Option]) -> ok
```

Types:

**Option** = {style, integer()}

See **external documentation**.

```
destroy(This::wxBufferedDC()) -> ok
```

Destroys this object, do not use object again

## wxBufferedPaintDC

---

Erlang module

See external documentation: **wxBufferedPaintDC**.

This class is derived (and can use functions) from:

*wxBufferedDC*

*wxMemoryDC*

*wxDC*

### DATA TYPES

`wxBufferedPaintDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new(Window::wxWindow() (see module wxWindow)) -> wxBufferedPaintDC()`**

Equivalent to `new(Window, [])`.

**`new(Window::wxWindow() (see module wxWindow), X::term()) -> wxBufferedPaintDC()`**

See **external documentation**.

Alternatives:

`new(Window::wxWindow:wxWindow(), Buffer::wxBitmap:wxBitmap()) -> new(Window, Buffer, [])`

`new(Window::wxWindow:wxWindow(), [Option]) -> wxBufferedPaintDC()`  
`Option = {style, integer()}`

**`new(Window::wxWindow() (see module wxWindow), Buffer::wxBitmap() (see module wxBitmap), Options::[Option]) -> wxBufferedPaintDC()`**

Types:

**`Option = {style, integer()}`**

See **external documentation**.

**`destroy(This::wxBufferedPaintDC()) -> ok`**

Destroys this object, do not use object again

# wxButton

Erlang module

See external documentation: **wxButton**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxButton()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxButton()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxButton()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxButton()`

Types:

**Option** = {label, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxButton(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, [])`.

`create(This::wxButton(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {label, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`getDefaultSize() -> {W::integer(), H::integer()}`

See external documentation.

## **wxButton**

---

**setDefault(This::wxButton()) -> ok**

See **external documentation**.

**setLabel(This::wxButton(), Label::string()) -> ok**

See **external documentation**.

**destroy(This::wxButton()) -> ok**

Destroys this object, do not use object again

# wxCalendarCtrl

Erlang module

See external documentation: **wxCalendarCtrl**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxCalendarCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxCalendarCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxCalendarCtrl()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxCalendarCtrl()`

Types:

**Option** = {date, datetime() (see module wx)} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxCalendarCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, [])`.

`create(This::wxCalendarCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {date, datetime() (see module wx)} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`setDate(This::wxCalendarCtrl(), Date::datetime() (see module wx)) -> bool()`

See external documentation.

`getDate(This::wxCalendarCtrl()) -> datetime()` (see module wx)

See external documentation.

`enableYearChange(This::wxCalendarCtrl()) -> ok`

Equivalent to `enableYearChange(This, [])`.

`enableYearChange(This::wxCalendarCtrl(), Options::[Option]) -> ok`

Types:

**Option** = {enable, bool()}

See external documentation.

`enableMonthChange(This::wxCalendarCtrl()) -> ok`

Equivalent to `enableMonthChange(This, [])`.

`enableMonthChange(This::wxCalendarCtrl(), Options::[Option]) -> ok`

Types:

**Option** = {enable, bool()}

See external documentation.

`enableHolidayDisplay(This::wxCalendarCtrl()) -> ok`

Equivalent to `enableHolidayDisplay(This, [])`.

`enableHolidayDisplay(This::wxCalendarCtrl(), Options::[Option]) -> ok`

Types:

**Option** = {display, bool()}

See external documentation.

`setHeaderColours(This::wxCalendarCtrl(), ColFg::colour() (see module wx), ColBg::colour() (see module wx)) -> ok`

See external documentation.

`getHeaderColourFg(This::wxCalendarCtrl()) -> colour()` (see module wx)

See external documentation.

`getHeaderColourBg(This::wxCalendarCtrl()) -> colour()` (see module wx)

See external documentation.

`setHighlightColours(This::wxCalendarCtrl(), ColFg::colour() (see module wx), ColBg::colour() (see module wx)) -> ok`

See external documentation.

`getHighlightColourFg(This::wxCalendarCtrl()) -> colour()` (see module wx)

See external documentation.

**getHighlightColourBg(This::wxCalendarCtrl()) -> colour()** (see module wx)

See external documentation.

**setHolidayColours(This::wxCalendarCtrl(), ColFg::colour() (see module wx), ColBg::colour() (see module wx)) -> ok**

See external documentation.

**getHolidayColourFg(This::wxCalendarCtrl()) -> colour()** (see module wx)

See external documentation.

**getHolidayColourBg(This::wxCalendarCtrl()) -> colour()** (see module wx)

See external documentation.

**getAttr(This::wxCalendarCtrl(), Day::integer()) -> wxCalendarDateAttr()** (see module wxCalendarDateAttr)

See external documentation.

**setAttr(This::wxCalendarCtrl(), Day::integer(), Attr::wxCalendarDateAttr() (see module wxCalendarDateAttr)) -> ok**

See external documentation.

**setHoliday(This::wxCalendarCtrl(), Day::integer()) -> ok**

See external documentation.

**resetAttr(This::wxCalendarCtrl(), Day::integer()) -> ok**

See external documentation.

**hitTest(This::wxCalendarCtrl(), Pos::{X::integer(), Y::integer()}) -> {WxCalendarHitTestResult, Date::datetime() (see module wx), Wd::WeekDay}**

Types:

**WxCalendarHitTestResult = integer()**

**WeekDay = integer()**

See external documentation.

WxCalendarHitTestResult is one of ?wxCAL\_HITTEST\_NOWHERE | ?wxCAL\_HITTEST\_HEADER | ?wxCAL\_HITTEST\_DAY | ?wxCAL\_HITTEST\_INCMONTH | ?wxCAL\_HITTEST\_DECMONTH | ?wxCAL\_HITTEST\_SURROUNDING\_WEEK

WeekDay is one of ?wxDateTime\_Sun | ?wxDateTime\_Mon | ?wxDateTime\_Tue | ?wxDateTime\_Wed | ?wxDateTime\_Thu | ?wxDateTime\_Fri | ?wxDateTime\_Sat | ?wxDateTime\_Inv\_WeekDay

**destroy(This::wxCalendarCtrl()) -> ok**

Destroys this object, do not use object again

## wxCalendarDateAttr

---

Erlang module

See external documentation: **wxCalendarDateAttr**.

### DATA TYPES

`wxCalendarDateAttr()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

**`new()` -> `wxCalendarDateAttr()`**

See external documentation.

**`new(X::WxCalendarDateBorder | term()) -> wxCalendarDateAttr()`**

See external documentation.

Alternatives:

`new(Border::WxCalendarDateBorder) -> new(Border, [])`

`new(ColText::wx:colour()) -> new(ColText, [])`

**`new(X::WxCalendarDateBorder | term(), Options::[Option]) -> wxCalendarDateAttr()`**

See external documentation.

Alternatives:

`new(Border::WxCalendarDateBorder, [Option]) -> wxCalendarDateAttr()`

Option = {colBorder, wx:colour()}

WxCalendarDateBorder = integer()

WxCalendarDateBorder is one of ?wxCAL\_BORDER\_NONE | ?wxCAL\_BORDER\_SQUARE | ?wxCAL\_BORDER\_ROUND

`new(ColText::wx:colour(), [Option]) -> wxCalendarDateAttr()`

Option = {colBack, wx:colour()} | {colBorder, wx:colour()} | {font, wxFont:wxFont()} | {border, WxCalendarDateBorder}

WxCalendarDateBorder = integer()

WxCalendarDateBorder is one of ?wxCAL\_BORDER\_NONE | ?wxCAL\_BORDER\_SQUARE | ?wxCAL\_BORDER\_ROUND

**`setTextColour(This::wxCalendarDateAttr(), ColText::colour() (see module wx)) -> ok`**

See external documentation.

**`setBackgroundColour(This::wxCalendarDateAttr(), ColBack::colour() (see module wx)) -> ok`**

See external documentation.



**setBorderColour**(This::wxCalendarDateAttr(), Col::colour() (see module wx)) -> ok

See external documentation.

**setFont**(This::wxCalendarDateAttr(), Font::wxFont() (see module wxFont)) -> ok

See external documentation.

**setBorder**(This::wxCalendarDateAttr(), Border::WxCalendarDateBorder) -> ok

Types:

WxCalendarDateBorder = integer()

See external documentation.

WxCalendarDateBorder is one of ?wxCAL\_BORDER\_NONE | ?wxCAL\_BORDER\_SQUARE | ?wxCAL\_BORDER\_ROUND

**setHoliday**(This::wxCalendarDateAttr(), Holiday::bool()) -> ok

See external documentation.

**hasTextColour**(This::wxCalendarDateAttr()) -> bool()

See external documentation.

**hasBackgroundColour**(This::wxCalendarDateAttr()) -> bool()

See external documentation.

**hasBorderColour**(This::wxCalendarDateAttr()) -> bool()

See external documentation.

**hasFont**(This::wxCalendarDateAttr()) -> bool()

See external documentation.

**hasBorder**(This::wxCalendarDateAttr()) -> bool()

See external documentation.

**isHoliday**(This::wxCalendarDateAttr()) -> bool()

See external documentation.

**getTextColour**(This::wxCalendarDateAttr()) -> colour() (see module wx)

See external documentation.

**getBackgroundColour**(This::wxCalendarDateAttr()) -> colour() (see module wx)

See external documentation.

**getBorderColour**(This::wxCalendarDateAttr()) -> colour() (see module wx)

See external documentation.

## **wxCalendarDateAttr**

---

**getFont(This::wxCalendarDateAttr()) -> wxFont()** (see module wxFont)

See **external documentation**.

**getBorder(This::wxCalendarDateAttr()) -> WxCalendarDateBorder**

Types:

**WxCalendarDateBorder = integer()**

See **external documentation**.

WxCalendarDateBorder is one of ?wxCAL\_BORDER\_NONE | ?wxCAL\_BORDER\_SQUARE | ?wxCAL\_BORDER\_ROUND

**destroy(This::wxCalendarDateAttr()) -> ok**

Destroys this object, do not use object again

# wxCalendarEvent

---

Erlang module

See external documentation: **wxCalendarEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*calendar\_sel\_changed*, *calendar\_day\_changed*, *calendar\_month\_changed*, *calendar\_year\_changed*,  
*calendar\_doubleclicked*, *calendar\_weekday\_clicked*

See also the message variant *#wxCalendar{}* event record type.

This class is derived (and can use functions) from:

*wxDateEvent*

*wxCommandEvent*

*wxEvent*

## DATA TYPES

`wxCalendarEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

**`getWeekDay(This::wxCalendarEvent()) -> WeekDay`**

Types:

**`WeekDay = integer()`**

See **external documentation**.

WeekDay is one of `?wxDateTime_Sun` | `?wxDateTime_Mon` | `?wxDateTime_Tue` | `?wxDateTime_Wed` | `?wxDateTime_Thu` | `?wxDateTime_Fri` | `?wxDateTime_Sat` | `?wxDateTime_Inv_WeekDay`

## wxCaret

---

Erlang module

See external documentation: **wxCaret**.

### DATA TYPES

`wxCaret()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

`new(Window::wxWindow() (see module wxWindow), Size::{W::integer(), H::integer()}) -> wxCaret()`

See external documentation.

`new(Window::wxWindow() (see module wxWindow), Width::integer(), Height::integer()) -> wxCaret()`

See external documentation.

`create(This::wxCaret(), Window::wxWindow() (see module wxWindow), Size::{W::integer(), H::integer()}) -> bool()`

See external documentation.

`create(This::wxCaret(), Window::wxWindow() (see module wxWindow), Width::integer(), Height::integer()) -> bool()`

See external documentation.

`getBlinkTime() -> integer()`

See external documentation.

`getPosition(This::wxCaret()) -> {X::integer(), Y::integer()}`

See external documentation.

`getSize(This::wxCaret()) -> {W::integer(), H::integer()}`

See external documentation.

`getWindow(This::wxCaret()) -> wxWindow() (see module wxWindow)`

See external documentation.

`hide(This::wxCaret()) -> ok`

See external documentation.

**isOk(This::wxCaret()) -> bool()**

See external documentation.

**isVisible(This::wxCaret()) -> bool()**

See external documentation.

**move(This::wxCaret(), Pt::{X::integer(), Y::integer()}) -> ok**

See external documentation.

**move(This::wxCaret(), X::integer(), Y::integer()) -> ok**

See external documentation.

**setBlinkTime(Milliseconds::integer()) -> ok**

See external documentation.

**setSize(This::wxCaret(), Size::{W::integer(), H::integer()}) -> ok**

See external documentation.

**setSize(This::wxCaret(), Width::integer(), Height::integer()) -> ok**

See external documentation.

**show(This::wxCaret()) -> ok**

Equivalent to *show(This, [])*.

**show(This::wxCaret(), Options::[Option]) -> ok**

Types:

**Option = {show, bool()}**

See external documentation.

**destroy(This::wxCaret()) -> ok**

Destroys this object, do not use object again

## wxCheckBox

---

Erlang module

See external documentation: **wxCheckBox**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxCheckBox()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxCheckBox()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string())  
-> wxCheckBox()`

Equivalent to `new(Parent, Id, Label, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string(),  
Options::[Option]) -> wxCheckBox()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} |  
{validator, wx()} (see module wx)

See external documentation.

`create(This::wxCheckBox(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string()) -> bool()`

Equivalent to `create(This, Parent, Id, Label, [])`.

`create(This::wxCheckBox(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} |  
{validator, wx()} (see module wx)

See external documentation.

`getValue(This::wxCheckBox()) -> bool()`

See external documentation.

**get3StateValue(This::wxCheckBox()) -> WxCheckBoxState**

Types:

**WxCheckBoxState = integer()**

See **external documentation**.

WxCheckBoxState is one of ?wxCHK\_UNCHECKED | ?wxCHK\_CHECKED | ?wxCHK\_UNDETERMINED

**is3rdStateAllowedForUser(This::wxCheckBox()) -> bool()**

See **external documentation**.

**is3State(This::wxCheckBox()) -> bool()**

See **external documentation**.

**isChecked(This::wxCheckBox()) -> bool()**

See **external documentation**.

**setValue(This::wxCheckBox(), State::bool()) -> ok**

See **external documentation**.

**set3StateValue(This::wxCheckBox(), State::WxCheckBoxState) -> ok**

Types:

**WxCheckBoxState = integer()**

See **external documentation**.

WxCheckBoxState is one of ?wxCHK\_UNCHECKED | ?wxCHK\_CHECKED | ?wxCHK\_UNDETERMINED

**destroy(This::wxCheckBox()) -> ok**

Destroys this object, do not use object again

## wxCheckListBox

---

Erlang module

See external documentation: **wxCheckListBox**.

This class is derived (and can use functions) from:

*wxListBox*

*wxControlWithItems*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxCheckListBox()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxCheckListBox()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxCheckListBox()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxCheckListBox()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {choices, [[string()]]} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`check(This::wxCheckListBox(), Index::integer()) -> ok`

Equivalent to `check(This, Index, [])`.

`check(This::wxCheckListBox(), Index::integer(), Options::[Option]) -> ok`

Types:

**Option** = {check, bool()}

See external documentation.

`isChecked(This::wxCheckListBox(), Index::integer()) -> bool()`

See external documentation.



**destroy(This::wxCheckListBox()) -> ok**

Destroys this object, do not use object again

## wxChildFocusEvent

---

Erlang module

See external documentation: **wxChildFocusEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*child\_focus*

See also the message variant *#wxChildFocus{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

### DATA TYPES

`wxChildFocusEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`getWindow(This::wxChildFocusEvent()) -> wxWindow()` (see module `wxWindow`)

See external documentation.

## wxChoice

Erlang module

See external documentation: **wxChoice**.

This class is derived (and can use functions) from:

*wxControlWithItems*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxChoice()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxChoice()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxChoice()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxChoice()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {choices, [[string()]]} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxChoice(), Parent::wxWindow() (see module wxWindow), Id::integer(), Pos::{X::integer(), Y::integer()}, Size::{W::integer(), H::integer()}, Choices::[[string()]]) -> bool()`

Equivalent to `create(This, Parent, Id, Pos, Size, Choices, [])`.

`create(This::wxChoice(), Parent::wxWindow() (see module wxWindow), Id::integer(), Pos::{X::integer(), Y::integer()}, Size::{W::integer(), H::integer()}, Choices::[[string()]], Options::[Option]) -> bool()`

Types:

**Option** = {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

**delete(This::wxChoice(), N::integer()) -> ok**

See **external documentation**.

**getColumns(This::wxChoice()) -> integer()**

See **external documentation**.

**setColumns(This::wxChoice()) -> ok**

Equivalent to *setColumns(This, [])*.

**setColumns(This::wxChoice(), Options::[Option]) -> ok**

Types:

**Option = {n, integer()}**

See **external documentation**.

**destroy(This::wxChoice()) -> ok**

Destroys this object, do not use object again

# wxChoicebook

---

Erlang module

See external documentation: **wxChoicebook**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxChoicebook()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxChoicebook()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxChoicebook()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxChoicebook()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`addPage(This::wxChoicebook(), Page::wxWindow() (see module wxWindow), Text::string()) -> bool()`

Equivalent to `addPage(This, Page, Text, [])`.

`addPage(This::wxChoicebook(), Page::wxWindow() (see module wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

**Option** = {bSelect, bool()} | {imageId, integer()}

See external documentation.

`advanceSelection(This::wxChoicebook()) -> ok`

Equivalent to `advanceSelection(This, [])`.

`advanceSelection(This::wxChoicebook(), Options::[Option]) -> ok`

Types:

`Option = {forward, bool()}`

See external documentation.

`assignImageList(This::wxChoicebook(), ImageList::wxImageList() (see module wxImageList)) -> ok`

See external documentation.

`create(This::wxChoicebook(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, [])`.

`create(This::wxChoicebook(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()`

Types:

`Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}`

See external documentation.

`deleteAllPages(This::wxChoicebook()) -> bool()`

See external documentation.

`deletePage(This::wxChoicebook(), N::integer()) -> bool()`

See external documentation.

`removePage(This::wxChoicebook(), N::integer()) -> bool()`

See external documentation.

`getCurrentPage(This::wxChoicebook()) -> wxWindow() (see module wxWindow)`

See external documentation.

`getImageList(This::wxChoicebook()) -> wxImageList() (see module wxImageList)`

See external documentation.

`getPage(This::wxChoicebook(), N::integer()) -> wxWindow() (see module wxWindow)`

See external documentation.

`getPageCount(This::wxChoicebook()) -> integer()`

See external documentation.

`getPageImage(This::wxChoicebook(), N::integer()) -> integer()`

See external documentation.

`getPageText(This::wxChoicebook(), N::integer()) -> string()`

See external documentation.

`getSelection(This::wxChoicebook()) -> integer()`

See external documentation.

`hitTest(This::wxChoicebook(), Pt::{X::integer(), Y::integer()}) -> {integer(), Flags::integer()}`

See external documentation.

`insertPage(This::wxChoicebook(), N::integer(), Page::wxWindow() (see module wxWindow), Text::string()) -> bool()`

Equivalent to `insertPage(This, N, Page, Text, [])`.

`insertPage(This::wxChoicebook(), N::integer(), Page::wxWindow() (see module wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

`Option = {bSelect, bool()} | {imageId, integer()}`

See external documentation.

`setImageList(This::wxChoicebook(), ImageList::wxImageList() (see module wxImageList)) -> ok`

See external documentation.

`setPageSize(This::wxChoicebook(), Size::{W::integer(), H::integer()}) -> ok`

See external documentation.

`setPageImage(This::wxChoicebook(), N::integer(), ImageId::integer()) -> bool()`

See external documentation.

`setPageText(This::wxChoicebook(), N::integer(), StrText::string()) -> bool()`

See external documentation.

`setSelection(This::wxChoicebook(), N::integer()) -> integer()`

See external documentation.

`changeSelection(This::wxChoicebook(), N::integer()) -> integer()`

See external documentation.

`destroy(This::wxChoicebook()) -> ok`

Destroys this object, do not use object again

## wxClientDC

---

Erlang module

See external documentation: **wxClientDC**.

This class is derived (and can use functions) from:

*wxWindowDC*

*wxDC*

## DATA TYPES

`wxClientDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

**`new()`** -> **`wxClientDC()`**

See **external documentation**.

**`new(Win::wxWindow())`** (see module **`wxWindow`**) -> **`wxClientDC()`**

See **external documentation**.

**`destroy(This::wxClientDC())`** -> **`ok`**

Destroys this object, do not use object again



# wxClipboard

---

Erlang module

See external documentation: **wxClipboard**.

## DATA TYPES

`wxClipboard()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxClipboard()`

See external documentation.

`addData(This::wxClipboard(), Data::wxDataObject() (see module wxDataObject))  
-> bool()`

See external documentation.

`clear(This::wxClipboard()) -> ok`

See external documentation.

`close(This::wxClipboard()) -> ok`

See external documentation.

`flush(This::wxClipboard()) -> bool()`

See external documentation.

`getData(This::wxClipboard(), Data::wxDataObject() (see module wxDataObject))  
-> bool()`

See external documentation.

`isOpen(This::wxClipboard()) -> bool()`

See external documentation.

`open(This::wxClipboard()) -> bool()`

See external documentation.

`setData(This::wxClipboard(), Data::wxDataObject() (see module wxDataObject))  
-> bool()`

See external documentation.

## wxClipboard

---

**usePrimarySelection(This::wxClipboard()) -> ok**

Equivalent to *usePrimarySelection(This, [])*.

**usePrimarySelection(This::wxClipboard(), Options::[Option]) -> ok**

Types:

**Option = {primary, bool()}**

See **external documentation**.

**isSupported(This::wxClipboard(), Format::integer()) -> bool()**

See **external documentation**.

**get() -> wxClipboard()**

See **external documentation**.

**destroy(This::wxClipboard()) -> ok**

Destroys this object, do not use object again

---

## wxCloseEvent

---

Erlang module

See external documentation: **wxCloseEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*close\_window, end\_session, query\_end\_session*

See also the message variant *#wxClose{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxCloseEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

`canVeto(This::wxCloseEvent()) -> bool()`

See external documentation.

`getLoggingOff(This::wxCloseEvent()) -> bool()`

See external documentation.

`setCanVeto(This::wxCloseEvent(), CanVeto::bool()) -> ok`

See external documentation.

`setLoggingOff(This::wxCloseEvent(), LogOff::bool()) -> ok`

See external documentation.

`veto(This::wxCloseEvent()) -> ok`

Equivalent to *veto(This, [])*.

`veto(This::wxCloseEvent(), Options::[Option]) -> ok`

Types:

**Option** = {veto, bool()}

See external documentation.

## wxColourData

---

Erlang module

See external documentation: **wxColourData**.

### DATA TYPES

`wxColourData()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxColourData()`

See external documentation.

`new(Data::wxColourData()) -> wxColourData()`

See external documentation.

`getChooseFull(This::wxColourData()) -> bool()`

See external documentation.

`getColour(This::wxColourData()) -> colour() (see module wx)`

See external documentation.

`getCustomColour(This::wxColourData(), I::integer()) -> colour() (see module wx)`

See external documentation.

`setChooseFull(This::wxColourData(), Flag::bool()) -> ok`

See external documentation.

`setColour(This::wxColourData(), Colour::colour() (see module wx)) -> ok`

See external documentation.

`setCustomColour(This::wxColourData(), I::integer(), Colour::colour() (see module wx)) -> ok`

See external documentation.

`destroy(This::wxColourData()) -> ok`

Destroys this object, do not use object again

## wxColourDialog

---

Erlang module

See external documentation: **wxColourDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxColourDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxColourDialog()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxColourDialog()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxColourDialog()`

Types:

**Option** = {data, wxColourData() (see module wxColourData)}

See external documentation.

`create(This::wxColourDialog(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

`create(This::wxColourDialog(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {data, wxColourData() (see module wxColourData)}

See external documentation.

`getColourData(This::wxColourDialog()) -> wxColourData() (see module wxColourData)`

See external documentation.

## **wxColourDialog**

---

**destroy(This::wxColourDialog()) -> ok**

Destroys this object, do not use object again

# wxColourPickerCtrl

Erlang module

See external documentation: **wxColourPickerCtrl**.

This class is derived (and can use functions) from:

*wxPickerBase*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxColourPickerCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxColourPickerCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxColourPickerCtrl()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxColourPickerCtrl()`

Types:

**Option** = {col, colour() (see module wx)} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxColourPickerCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, [])`.

`create(This::wxColourPickerCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {col, colour() (see module wx)} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

**getColour(This::wxColourPickerCtrl()) -> colour()** (see module wx)

See **external documentation**.

**setColour(This::wxColourPickerCtrl(), X::string() | term()) -> bool() | ok**

See **external documentation**.

Alternatives:

**setColour(This::wxColourPickerCtrl(), Text::string()) -> bool()**

**setColour(This::wxColourPickerCtrl(), Col::wx:colour()) -> ok**

**destroy(This::wxColourPickerCtrl()) -> ok**

Destroys this object, do not use object again



## wxColourPickerEvent

---

Erlang module

See external documentation: **wxColourPickerEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*command\_colourpicker\_changed*

See also the message variant *#wxColourPicker{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

### DATA TYPES

`wxColourPickerEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`getColour(This::wxColourPickerEvent()) -> colour()` (see module `wx`)

See **external documentation**.

## wxComboBox

---

Erlang module

See external documentation: **wxComboBox**.

This class is derived (and can use functions) from:

*wxControlWithItems*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxComboBox()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxComboBox()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxComboBox()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxComboBox()`

Types:

**Option** = {value, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {choices, [[string()]]} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxComboBox(), Parent::wxWindow() (see module wxWindow), Id::integer(), Value::string(), Pos::{X::integer(), Y::integer()}, Size::{W::integer(), H::integer()}, Choices::[[string()]]) -> bool()`

Equivalent to `create(This, Parent, Id, Value, Pos, Size, Choices, [])`.

`create(This::wxComboBox(), Parent::wxWindow() (see module wxWindow), Id::integer(), Value::string(), Pos::{X::integer(), Y::integer()}, Size::{W::integer(), H::integer()}, Choices::[[string()]], Options::[Option]) -> bool()`

Types:

**Option** = {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`canCopy(This::wxComboBox()) -> bool()`

See external documentation.

`canCut(This::wxComboBox()) -> bool()`

See external documentation.

`canPaste(This::wxComboBox()) -> bool()`

See external documentation.

`canRedo(This::wxComboBox()) -> bool()`

See external documentation.

`canUndo(This::wxComboBox()) -> bool()`

See external documentation.

`copy(This::wxComboBox()) -> ok`

See external documentation.

`cut(This::wxComboBox()) -> ok`

See external documentation.

`getInsertionPoint(This::wxComboBox()) -> integer()`

See external documentation.

`getLastPosition(This::wxComboBox()) -> integer()`

See external documentation.

`getValue(This::wxComboBox()) -> string()`

See external documentation.

`paste(This::wxComboBox()) -> ok`

See external documentation.

`redo(This::wxComboBox()) -> ok`

See external documentation.

`replace(This::wxComboBox(), From::integer(), To::integer(), Value::string())  
-> ok`

See external documentation.

`remove(This::wxComboBox(), From::integer(), To::integer()) -> ok`

See external documentation.

`setInsertionPoint(This::wxComboBox(), Pos::integer()) -> ok`

See [external documentation](#).

`setInsertionPointEnd(This::wxComboBox()) -> ok`

See [external documentation](#).

`setSelection(This::wxComboBox(), N::integer()) -> ok`

See [external documentation](#).

`setSelection(This::wxComboBox(), From::integer(), To::integer()) -> ok`

See [external documentation](#).

`setValue(This::wxComboBox(), Value::string()) -> ok`

See [external documentation](#).

`undo(This::wxComboBox()) -> ok`

See [external documentation](#).

`destroy(This::wxComboBox()) -> ok`

Destroys this object, do not use object again

# wxCommandEvent

---

Erlang module

See external documentation: **wxCommandEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*command\_button\_clicked, command\_checkbox\_clicked, command\_choice\_selected, command\_listbox\_selected, command\_listbox\_doubleclicked, command\_text\_updated, command\_text\_enter, command\_menu\_selected, command\_slider\_updated, command\_radiobox\_selected, command\_radiobutton\_selected, command\_scrollbar\_updated, command\_vlbox\_selected, command\_combobox\_selected, command\_tool\_rclicked, command\_tool\_enter, command\_checklistbox\_toggled, command\_togglebutton\_clicked, command\_left\_click, command\_left\_dclick, command\_right\_click, command\_set\_focus, command\_kill\_focus, command\_enter*

See also the message variant *#wxCommandEvent* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

`wxCommandEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getClientData(This::wxCommandEvent()) -> term()`

See external documentation.

`getExtraLong(This::wxCommandEvent()) -> integer()`

See external documentation.

`getInt(This::wxCommandEvent()) -> integer()`

See external documentation.

`getSelection(This::wxCommandEvent()) -> integer()`

See external documentation.

`getString(This::wxCommandEvent()) -> string()`

See external documentation.

`isChecked(This::wxCommandEvent()) -> bool()`

See external documentation.

## **wxCommandEvent**

---

**isSelection(This::wxCommandEvent()) -> bool()**

See **external documentation**.

**setInt(This::wxCommandEvent(), I::integer()) -> ok**

See **external documentation**.

**setString(This::wxCommandEvent(), S::string()) -> ok**

See **external documentation**.

## wxContextMenuEvent

---

Erlang module

See external documentation: **wxContextMenuEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*context\_menu*

See also the message variant *#wxContextMenu{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvt*

## DATA TYPES

`wxContextMenuEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getPosition(This::wxContextMenuEvent()) -> {X::integer(), Y::integer()}`

See external documentation.

`setPosition(This::wxContextMenuEvent(), Pos::{X::integer(), Y::integer()}) -> ok`

See external documentation.

## wxControl

---

Erlang module

See external documentation: **wxControl**.

This class is derived (and can use functions) from:

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxControl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getLabel(This::wxControl()) -> string()`

See **external documentation**.

`setLabel(This::wxControl(), Label::string()) -> ok`

See **external documentation**.



## wxControlWithItems

---

Erlang module

See external documentation: **wxControlWithItems**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxControlWithItems()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`append(This::wxControlWithItems(), Item::string()) -> integer()`

See external documentation.

`append(This::wxControlWithItems(), Item::string(), ClientData::term()) -> integer()`

See external documentation.

`appendStrings(This::wxControlWithItems(), Strings:[[string()]]) -> ok`

See external documentation.

`clear(This::wxControlWithItems()) -> ok`

See external documentation.

`delete(This::wxControlWithItems(), N::integer()) -> ok`

See external documentation.

`findString(This::wxControlWithItems(), S::string()) -> integer()`

Equivalent to `findString(This, S, [])`.

`findString(This::wxControlWithItems(), S::string(), Options::[Option]) -> integer()`

Types:

**Option** = {bCase, bool()}

See external documentation.

`getClientData(This::wxControlWithItems(), N::integer()) -> term()`

See external documentation.

```
setClientData(This::wxControlWithItems(), N::integer(), ClientData::term()) -> ok
```

See external documentation.

```
getCount(This::wxControlWithItems()) -> integer()
```

See external documentation.

```
getSelection(This::wxControlWithItems()) -> integer()
```

See external documentation.

```
getString(This::wxControlWithItems(), N::integer()) -> string()
```

See external documentation.

```
getStringSelection(This::wxControlWithItems()) -> string()
```

See external documentation.

```
insert(This::wxControlWithItems(), Item::string(), Pos::integer()) -> integer()
```

See external documentation.

```
insert(This::wxControlWithItems(), Item::string(), Pos::integer(), ClientData::term()) -> integer()
```

See external documentation.

```
isEmpty(This::wxControlWithItems()) -> bool()
```

See external documentation.

```
select(This::wxControlWithItems(), N::integer()) -> ok
```

See external documentation.

```
setSelection(This::wxControlWithItems(), N::integer()) -> ok
```

See external documentation.

```
setString(This::wxControlWithItems(), N::integer(), S::string()) -> ok
```

See external documentation.

```
setStringSelection(This::wxControlWithItems(), S::string()) -> bool()
```

See external documentation.

## wxCursor

---

Erlang module

See external documentation: **wxCursor**.

This class is derived (and can use functions) from:  
*wxBitmap*

## DATA TYPES

`wxCursor()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxCursor()`

See external documentation.

`new(X::integer() | term()) -> wxCursor()`

See external documentation.

Alternatives:

`new(CursorId::integer()) -> wxCursor()`

`new(Image::wxImage:wxImage()) -> wxCursor()`

`new(Bits::binary(), Width::integer(), Height::integer()) -> wxCursor()`

Equivalent to `new(Bits, Width, Height, [])`.

`new(Bits::binary(), Width::integer(), Height::integer(), Options::[Option]) -> wxCursor()`

Types:

**Option** = {hotSpotX, integer()} | {hotSpotY, integer()}

See external documentation.

`ok(This::wxCursor()) -> bool()`

See external documentation.

`destroy(This::wxCursor()) -> ok`

Destroys this object, do not use object again

## wxDC

---

Erlang module

See external documentation: **wxDC**.

### DATA TYPES

`wxDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

```
blit(This::wxDC(), DestPt::{X::integer(), Y::integer()}, Sz::{W::integer(),  
H::integer()}, Source::wxDC(), SrcPt::{X::integer(), Y::integer()}) -> bool()
```

Equivalent to *blit(This, DestPt, Sz, Source, SrcPt, [])*.

```
blit(This::wxDC(), DestPt::{X::integer(), Y::integer()}, Sz::{W::integer(),  
H::integer()}, Source::wxDC(), SrcPt::{X::integer(), Y::integer()}, Options::  
[Option]) -> bool()
```

Types:

**Option** = {**rop**, **integer()**} | {**useMask**, **bool()**} | {**srcPtMask**, {**X::integer()**, **Y::integer()**}}

See external documentation.

```
calcBoundingBox(This::wxDC(), X::integer(), Y::integer()) -> ok
```

See external documentation.

```
clear(This::wxDC()) -> ok
```

See external documentation.

```
computeScaleAndOrigin(This::wxDC()) -> ok
```

See external documentation.

```
crossHair(This::wxDC(), Pt::{X::integer(), Y::integer()}) -> ok
```

See external documentation.

```
destroyClippingRegion(This::wxDC()) -> ok
```

See external documentation.

```
deviceToLogicalX(This::wxDC(), X::integer()) -> integer()
```

See external documentation.

`deviceToLogicalXRel(This::wxDC(), X::integer()) -> integer()`

See external documentation.

`deviceToLogicalY(This::wxDC(), Y::integer()) -> integer()`

See external documentation.

`deviceToLogicalYRel(This::wxDC(), Y::integer()) -> integer()`

See external documentation.

`drawArc(This::wxDC(), Pt1::{X::integer(), Y::integer()}, Pt2::{X::integer(), Y::integer()}, Centre::{X::integer(), Y::integer()}) -> ok`

See external documentation.

`drawBitmap(This::wxDC(), Bmp::wxBitmap() (see module wxBitmap), Pt::{X::integer(), Y::integer()}) -> ok`

Equivalent to `drawBitmap(This, Bmp, Pt, [])`.

`drawBitmap(This::wxDC(), Bmp::wxBitmap() (see module wxBitmap), Pt::{X::integer(), Y::integer()}, Options::[Option]) -> ok`

Types:

`Option = {useMask, bool()}`

See external documentation.

`drawCheckMark(This::wxDC(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> ok`

See external documentation.

`drawCircle(This::wxDC(), Pt::{X::integer(), Y::integer()}, Radius::integer()) -> ok`

See external documentation.

`drawEllipse(This::wxDC(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> ok`

See external documentation.

`drawEllipse(This::wxDC(), Pt::{X::integer(), Y::integer()}, Sz::{W::integer(), H::integer()}) -> ok`

See external documentation.

`drawEllipticArc(This::wxDC(), Pt::{X::integer(), Y::integer()}, Sz::{W::integer(), H::integer()}, Sa::float(), Ea::float()) -> ok`

See external documentation.

```
drawIcon(This::wxDC(), Icon::wxIcon() (see module wxIcon), Pt::{X::integer(),  
Y::integer()}) -> ok
```

See external documentation.

```
drawLabel(This::wxDC(), Text::string(), Rect::{X::integer(), Y::integer(),  
W::integer(), H::integer()}) -> ok
```

Equivalent to *drawLabel(This, Text, Rect, [])*.

```
drawLabel(This::wxDC(), Text::string(), Rect::{X::integer(), Y::integer(),  
W::integer(), H::integer()}, Options::[Option]) -> ok
```

Types:

**Option** = {alignment, integer()} | {indexAccel, integer()}

See external documentation.

```
drawLine(This::wxDC(), Pt1::{X::integer(), Y::integer()}, Pt2::{X::integer(),  
Y::integer()}) -> ok
```

See external documentation.

```
drawLines(This::wxDC(), Points::{[X::integer(), Y::integer()]}) -> ok
```

Equivalent to *drawLines(This, Points, [])*.

```
drawLines(This::wxDC(), Points::{[X::integer(), Y::integer()]}, Options::  
[Option]) -> ok
```

Types:

**Option** = {xoffset, integer()} | {yoffset, integer()}

See external documentation.

```
drawPolygon(This::wxDC(), Points::{[X::integer(), Y::integer()]}) -> ok
```

Equivalent to *drawPolygon(This, Points, [])*.

```
drawPolygon(This::wxDC(), Points::{[X::integer(), Y::integer()]}, Options::  
[Option]) -> ok
```

Types:

**Option** = {xoffset, integer()} | {yoffset, integer()} | {fillStyle, integer()}

See external documentation.

```
drawPoint(This::wxDC(), Pt::{X::integer(), Y::integer()}) -> ok
```

See external documentation.

```
drawRectangle(This::wxDC(), Rect::{X::integer(), Y::integer(), W::integer(),  
H::integer()}) -> ok
```

See external documentation.

```
drawRectangle(This::wxDC(), Pt::{X::integer(), Y::integer()}, Sz::
{W::integer(), H::integer()}) -> ok
```

See external documentation.

```
drawRotatedText(This::wxDC(), Text::string(), Pt::{X::integer(),
Y::integer()}, Angle::float()) -> ok
```

See external documentation.

```
drawRoundedRectangle(This::wxDC(), R::{X::integer(), Y::integer(),
W::integer(), H::integer()}, Radius::float()) -> ok
```

See external documentation.

```
drawRoundedRectangle(This::wxDC(), Pt::{X::integer(), Y::integer()}, Sz::
{W::integer(), H::integer()}, Radius::float()) -> ok
```

See external documentation.

```
drawText(This::wxDC(), Text::string(), Pt::{X::integer(), Y::integer()}) ->
ok
```

See external documentation.

```
endDoc(This::wxDC()) -> ok
```

See external documentation.

```
endPage(This::wxDC()) -> ok
```

See external documentation.

```
floodFill(This::wxDC(), Pt::{X::integer(), Y::integer()}, Col::colour() (see
module wx)) -> bool()
```

Equivalent to *floodFill(This, Pt, Col, [])*.

```
floodFill(This::wxDC(), Pt::{X::integer(), Y::integer()}, Col::colour() (see
module wx), Options::[Option]) -> bool()
```

Types:

**Option** = {style, integer()}

See external documentation.

```
getBackground(This::wxDC()) -> wxBrush() (see module wxBrush)
```

See external documentation.

```
getBackgroundMode(This::wxDC()) -> integer()
```

See external documentation.

```
getBrush(This::wxDC()) -> wxBrush() (see module wxBrush)
```

See external documentation.

**getCharHeight(This::wxDC()) -> integer()**

See external documentation.

**getCharWidth(This::wxDC()) -> integer()**

See external documentation.

**getClippingBox(This::wxDC(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> ok**

See external documentation.

**getFont(This::wxDC()) -> wxFont() (see module wxFont)**

See external documentation.

**getLayoutDirection(This::wxDC()) -> WxLayoutDirection**

Types:

**WxLayoutDirection = integer()**

See external documentation.

WxLayoutDirection is one of ?wxLayout\_Default | ?wxLayout\_LeftToRight | ?wxLayout\_RightToLeft

**getLogicalFunction(This::wxDC()) -> integer()**

See external documentation.

**getMapMode(This::wxDC()) -> integer()**

See external documentation.

**getMultiLineTextExtent(This::wxDC(), String::string()) -> {W::integer(), H::integer()}**

See external documentation.

**getMultiLineTextExtent(This::wxDC(), String::string(), Options::[Option]) -> {Width::integer(), Height::integer(), HeightLine::integer()}**

Types:

**Option = {font, wxFont() (see module wxFont)}**

See external documentation.

**getPartialTextExtents(This::wxDC(), Text::string(), Widths::[integer()]) -> bool()**

See external documentation.

**getPen(This::wxDC()) -> wxPen() (see module wxPen)**

See external documentation.



`getPixel(This::wxDC(), Pt::{X::integer(), Y::integer()}, Col::colour() (see module wx)) -> bool()`

See external documentation.

`getPPI(This::wxDC()) -> {W::integer(), H::integer()}`

See external documentation.

`getSize(This::wxDC()) -> {W::integer(), H::integer()}`

See external documentation.

`getSizeMM(This::wxDC()) -> {W::integer(), H::integer()}`

See external documentation.

`getTextBackground(This::wxDC()) -> colour() (see module wx)`

See external documentation.

`getTextExtent(This::wxDC(), String::string()) -> {W::integer(), H::integer()}`

See external documentation.

`getTextExtent(This::wxDC(), String::string(), Options::[Option]) -> {X::integer(), Y::integer(), Descent::integer(), ExternalLeading::integer()}`

Types:

`Option = {theFont, wxFont() (see module wxFont)}`

See external documentation.

`getTextForeground(This::wxDC()) -> colour() (see module wx)`

See external documentation.

`getUserScale(This::wxDC()) -> {X::float(), Y::float()}`

See external documentation.

`gradientFillConcentric(This::wxDC(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}, InitialColour::colour() (see module wx), DestColour::colour() (see module wx)) -> ok`

See external documentation.

`gradientFillConcentric(This::wxDC(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}, InitialColour::colour() (see module wx), DestColour::colour() (see module wx), CircleCenter::{X::integer(), Y::integer()}) -> ok`

See external documentation.

```
gradientFillLinear(This::wxDC(), Rect::{X::integer(), Y::integer(),  
W::integer(), H::integer()}, InitialColour::colour() (see module wx),  
DestColour::colour() (see module wx)) -> ok
```

Equivalent to *gradientFillLinear(This, Rect, InitialColour, DestColour, [])*.

```
gradientFillLinear(This::wxDC(), Rect::{X::integer(), Y::integer(),  
W::integer(), H::integer()}, InitialColour::colour() (see module wx),  
DestColour::colour() (see module wx), Options::[Option]) -> ok
```

Types:

**Option** = {nDirection, WxDirection}

**WxDirection** = integer()

See **external documentation**.

WxDirection is one of ?wxLEFT | ?wxRIGHT | ?wxUP | ?wxDOWN | ?wxTOP | ?wxBOTTOM | ?wxNORTH | ?  
wxSOUTH | ?wxWEST | ?wxEAST | ?wxALL

```
logicalToDeviceX(This::wxDC(), X::integer()) -> integer()
```

See **external documentation**.

```
logicalToDeviceXRel(This::wxDC(), X::integer()) -> integer()
```

See **external documentation**.

```
logicalToDeviceY(This::wxDC(), Y::integer()) -> integer()
```

See **external documentation**.

```
logicalToDeviceYRel(This::wxDC(), Y::integer()) -> integer()
```

See **external documentation**.

```
maxX(This::wxDC()) -> integer()
```

See **external documentation**.

```
maxY(This::wxDC()) -> integer()
```

See **external documentation**.

```
minX(This::wxDC()) -> integer()
```

See **external documentation**.

```
minY(This::wxDC()) -> integer()
```

See **external documentation**.

```
isOk(This::wxDC()) -> bool()
```

See **external documentation**.

```
resetBoundingBox(This::wxDC()) -> ok
```

See **external documentation**.

**setAxisOrientation(This::wxDC(), XLeftRight::bool(), YBottomUp::bool()) -> ok**  
See external documentation.

**setBackground(This::wxDC(), Brush::wxBrush() (see module wxBrush)) -> ok**  
See external documentation.

**setBackgroundMode(This::wxDC(), Mode::integer()) -> ok**  
See external documentation.

**setBrush(This::wxDC(), Brush::wxBrush() (see module wxBrush)) -> ok**  
See external documentation.

**setClippingRegion(This::wxDC(), X::term()) -> ok**  
See external documentation.  
Alternatives:

**setClippingRegion(This::wxDC(), Region::wxRegion:wxRegion()) -> ok**  
**setClippingRegion(This::wxDC(), Rect::**  
**{X::integer(),Y::integer(),W::integer(),H::integer()}) -> ok**

**setClippingRegion(This::wxDC(), Pt::{X::integer(), Y::integer()}, Sz::**  
**{W::integer(), H::integer()}) -> ok**  
See external documentation.

**setDeviceOrigin(This::wxDC(), X::integer(), Y::integer()) -> ok**  
See external documentation.

**setFont(This::wxDC(), Font::wxFont() (see module wxFont)) -> ok**  
See external documentation.

**setLayoutDirection(This::wxDC(), Dir::WxLayoutDirection) -> ok**  
Types:

**WxLayoutDirection = integer()**

See external documentation.  
WxLayoutDirection is one of ?wxLayout\_Default | ?wxLayout\_LeftToRight | ?wxLayout\_RightToLeft

**setLogicalFunction(This::wxDC(), Function::integer()) -> ok**  
See external documentation.

**setMapMode(This::wxDC(), Mode::integer()) -> ok**  
See external documentation.

**setPalette(This::wxDC(), Palette::wxPalette() (see module wxPalette)) -> ok**  
See external documentation.

`setPen(This::wxDC(), Pen::wxPen() (see module wxPen)) -> ok`

See external documentation.

`setTextBackground(This::wxDC(), Colour::colour() (see module wx)) -> ok`

See external documentation.

`setTextForeground(This::wxDC(), Colour::colour() (see module wx)) -> ok`

See external documentation.

`setUserScale(This::wxDC(), X::float(), Y::float()) -> ok`

See external documentation.

`startDoc(This::wxDC(), Message::string()) -> bool()`

See external documentation.

`startPage(This::wxDC()) -> ok`

See external documentation.

## wxDataObject

---

Erlang module

See external documentation: **wxDataObject**.

### DATA TYPES

`wxDataObject()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxDatEvent

---

Erlang module

See external documentation: **wxDatEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*date\_changed*

See also the message variant *#wxDat{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvt*

## DATA TYPES

*wxDatEvent()*

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**getDate(This::wxDatEvent()) -> datetime()** (see module **wx**)

See external documentation.

## wxDatePickerCtrl

---

Erlang module

See external documentation: **wxDatePickerCtrl**.

This class is derived (and can use functions) from:

*wxPickerBase*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxDatePickerCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxDatePickerCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxDatePickerCtrl()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxDatePickerCtrl()`

Types:

**Option** = {date, datetime() (see module wx)} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`getRange(This::wxDatePickerCtrl(), Dt1::datetime() (see module wx), Dt2::datetime() (see module wx)) -> bool()`

See external documentation.

`getValue(This::wxDatePickerCtrl()) -> datetime() (see module wx)`

See external documentation.

`setRange(This::wxDatePickerCtrl(), Dt1::datetime() (see module wx), Dt2::datetime() (see module wx)) -> ok`

See external documentation.

## **wxDaterangepickerCtrl**

---

**setValue(This::wxDaterangepickerCtrl(), Date::datetime() (see module wx)) -> ok**

See **external documentation**.

**destroy(This::wxDaterangepickerCtrl()) -> ok**

Destroys this object, do not use object again



# wxDialog

---

Erlang module

See external documentation: **wxDialog**.

This class is derived (and can use functions) from:

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxDialog()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string())  
-> wxDialog()`

Equivalent to `new(Parent, Id, Title, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string(),  
Options::[Option]) -> wxDialog()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxDialog(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Title::string()) -> bool()`

Equivalent to `create(This, Parent, Id, Title, [])`.

`create(This::wxDialog(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Title::string(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`createButtonSizer(This::wxDialog(), Flags::integer()) -> wxSizer() (see  
module wxSizer)`

See external documentation.

`createStdDialogButtonSizer(This::wxDialog(), Flags::integer()) -> wxStdDialogButtonSizer()` (see module `wxStdDialogButtonSizer`)

See external documentation.

`endModal(This::wxDialog(), RetCode::integer()) -> ok`

See external documentation.

`getAffirmativeId(This::wxDialog()) -> integer()`

See external documentation.

`getReturnCode(This::wxDialog()) -> integer()`

See external documentation.

`isModal(This::wxDialog()) -> bool()`

See external documentation.

`setAffirmativeId(This::wxDialog(), AffirmativeId::integer()) -> ok`

See external documentation.

`setReturnCode(This::wxDialog(), ReturnCode::integer()) -> ok`

See external documentation.

`show(This::wxDialog()) -> bool()`

Equivalent to `show(This, [])`.

`show(This::wxDialog(), Options::[Option]) -> bool()`

Types:

**Option** = {show, bool()}

See external documentation.

`showModal(This::wxDialog()) -> integer()`

See external documentation.

`destroy(This::wxDialog()) -> ok`

Destroys this object, do not use object again

## wxDirDialog

---

Erlang module

See external documentation: **wxDirDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxDirDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new(Parent::wxWindow() (see module wxWindow)) -> wxDirDialog()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxDirDialog()`

Types:

**Option** = {title, string()} | {defaultPath, string()} | {style, integer()} | {pos, {X::integer(), Y::integer()}} | {sz, {W::integer(), H::integer()}}

See external documentation.

`getPath(This::wxDirDialog()) -> string()`

See external documentation.

`getMessage(This::wxDirDialog()) -> string()`

See external documentation.

`setMessage(This::wxDirDialog(), Message::string()) -> ok`

See external documentation.

`setPath(This::wxDirDialog(), Path::string()) -> ok`

See external documentation.

`destroy(This::wxDirDialog()) -> ok`

Destroys this object, do not use object again

## wxDirPickerCtrl

---

Erlang module

See external documentation: **wxDirPickerCtrl**.

This class is derived (and can use functions) from:

*wxPickerBase*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxDirPickerCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxDirPickerCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxDirPickerCtrl()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxDirPickerCtrl()`

Types:

**Option** = {path, string()} | {message, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxDirPickerCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, [])`.

`create(This::wxDirPickerCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {path, string()} | {message, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

**getPath(This::wxDirPickerCtrl()) -> string()**

See external documentation.

**setPath(This::wxDirPickerCtrl(), Str::string()) -> ok**

See external documentation.

**destroy(This::wxDirPickerCtrl()) -> ok**

Destroys this object, do not use object again

## wxDisplayChangedEvent

---

Erlang module

See external documentation: **wxDisplayChangedEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*display\_changed*

See also the message variant *#wxDisplayChanged{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxDisplayChangedEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxEraseEvent

---

Erlang module

See external documentation: **wxEraseEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*erase\_background*

See also the message variant *#wxErase{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxEraseEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`getDC(This::wxEraseEvent()) -> wxDC()` (see module `wxDC`)

See **external documentation**.

## wxEvent

---

Erlang module

See external documentation: **wxEvent**.

### DATA TYPES

`wxEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`getId(This::wxEvent()) -> integer()`

See external documentation.

`getSkipped(This::wxEvent()) -> bool()`

See external documentation.

`getTimestamp(This::wxEvent()) -> integer()`

See external documentation.

`isCommandEvent(This::wxEvent()) -> bool()`

See external documentation.

`resumePropagation(This::wxEvent(), PropagationLevel::integer()) -> ok`

See external documentation.

`shouldPropagate(This::wxEvent()) -> bool()`

See external documentation.

`skip(This::wxEvent()) -> ok`

Equivalent to `skip(This, [])`.

`skip(This::wxEvent(), Options::[Option]) -> ok`

Types:

**Option** = {skip, bool()}

See external documentation.

`stopPropagation(This::wxEvent()) -> integer()`

See external documentation.



## wxEvtHandler

Erlang module

The Event handler.

To get events from wxwidgets objects you subscribe to them by calling `connect/[2-3]`. Events are sent as messages, if no callback was supplied These messages will be `#wx{}` where `EventRecord` is a record that depends on the *event type*. The records are defined in: `wx/include/wx.hrl`.

If a callback was supplied to `connect`, the callback will be invoked (in another process) to handle the event. The callback should be of arity 2. `fun(EventRecord::wx(), EventObject::wxObject())`.

Beware that the callback will be in executed in new process each time.

**The original documentation.**

### DATA TYPES

```
wx() = #wx{id=integer(), obj=wxObject() (see module wx), userData=term(),
event=Rec}
```

Rec is a event record.

```
wxAuiManager() = #wxAuiManager{type=wxEventType(), manager=wxAuiManager()
(see module wxAuiManager), pane=wxAuiPaneInfo() (see module wxAuiPaneInfo),
button=integer(), veto_flag=bool(), canveto_flag=bool(), dc=wxDC() (see
module wxDC)}
```

EventType:

*au\_i\_pane\_button, au\_i\_pane\_close, au\_i\_pane\_maximize, au\_i\_pane\_restore, au\_i\_render, au\_i\_find\_manager*

Callback event: *wxAuiManagerEvent*

```
wxAuiNotebook() = #wxAuiNotebook{type=wxEventType(), old_selection=integer(),
selection=integer(), drag_source=wxAuiNotebook() (see module wxAuiNotebook)}
```

EventType:

*command\_aui\_notebook\_page\_close, command\_aui\_notebook\_page\_changed,*  
*command\_aui\_notebook\_page\_changing, command\_aui\_notebook\_button, command\_aui\_notebook\_begin\_drag,*  
*command\_aui\_notebook\_end\_drag, command\_aui\_notebook\_drag\_motion, command\_aui\_notebook\_allow\_dnd,*  
*command\_aui\_notebook\_tab\_middle\_down, command\_aui\_notebook\_tab\_middle\_up,*  
*command\_aui\_notebook\_tab\_right\_down, command\_aui\_notebook\_tab\_right\_up,*  
*command\_aui\_notebook\_page\_closed, command\_aui\_notebook\_drag\_done, command\_aui\_notebook\_bg\_dclick*

Callback event: *wxAuiNotebookEvent*

```
wxCalendar() = #wxCalendar{type=wxEventType() }
```

EventType:

*calendar\_sel\_changed, calendar\_day\_changed, calendar\_month\_changed, calendar\_year\_changed,*  
*calendar\_doubleclicked, calendar\_weekday\_clicked*

Callback event: *wxCalendarEvent*

```
wxChildFocus() = #wxChildFocus{type=wxEventType() }
```

EventType:

*child\_focus*

Callback event: *wxChildFocusEvent*

`wxClose()` = `#wxClose{type=wxEventType() }`

EventType:

*close\_window, end\_session, query\_end\_session*

Callback event: *wxCloseEvent*

`wxColourPicker()` = `#wxColourPicker{type=wxEventType(), colour=colour() (see module wx)}`

EventType:

*command\_colourpicker\_changed*

Callback event: *wxColourPickerEvent*

`wxCommand()` = `#wxCommand{type=wxEventType(), cmdString=string(), commandInt=integer(), extraLong=integer() }`

EventType:

*command\_button\_clicked, command\_checkbox\_clicked, command\_choice\_selected, command\_listbox\_selected, command\_listbox\_doubleclicked, command\_text\_updated, command\_text\_enter, command\_menu\_selected, command\_slider\_updated, command\_radiobox\_selected, command\_radiobutton\_selected, command\_scrollbar\_updated, command\_vlbox\_selected, command\_combobox\_selected, command\_tool\_rclicked, command\_tool\_enter, command\_checklistbox\_toggled, command\_togglebutton\_clicked, command\_left\_click, command\_left\_dclick, command\_right\_click, command\_set\_focus, command\_kill\_focus, command\_enter*

Callback event: *wxCommandEvent*

`wxContextMenu()` = `#wxContextMenu{type=wxEventType() }`

EventType:

*context\_menu*

Callback event: *wxContextMenuEvent*

`wxDate()` = `#wxDate{type=wxEventType(), date=datetime() (see module wx)}`

EventType:

*date\_changed*

Callback event: *wxDateEvent*

`wxDisplayChanged()` = `#wxDisplayChanged{type=wxEventType() }`

EventType:

*display\_changed*

Callback event: *wxDisplayChangedEvent*

`wxErase()` = `#wxErase{type=wxEventType(), dc=wxDC() (see module wxDC)}`

EventType:

*erase\_background*

Callback event: *wxEraseEvent*

`wxEventType()` = `aiui_find_manager | aiui_pane_button | aiui_pane_close | aiui_pane_maximize | aiui_pane_restore | aiui_render | calendar_day_changed | calendar_doubleclicked | calendar_month_changed | calendar_sel_changed | calendar_weekday_clicked | calendar_year_changed | char | char_hook`

| child\_focus | close\_window | command\_auinotebook\_allow\_dnd |  
command\_auinotebook\_begin\_drag | command\_auinotebook\_bg\_dclick  
| command\_auinotebook\_button | command\_auinotebook\_drag\_done |  
command\_auinotebook\_drag\_motion | command\_auinotebook\_end\_drag |  
command\_auinotebook\_page\_changed | command\_auinotebook\_page\_changing  
| command\_auinotebook\_page\_close | command\_auinotebook\_page\_closed |  
command\_auinotebook\_tab\_middle\_down | command\_auinotebook\_tab\_middle\_up  
| command\_auinotebook\_tab\_right\_down | command\_auinotebook\_tab\_right\_up  
| command\_button\_clicked | command\_checkbox\_clicked |  
command\_checklistbox\_toggled | command\_choice\_selected |  
command\_colourpicker\_changed | command\_combobox\_selected |  
command\_dirpicker\_changed | command\_enter | command\_filepicker\_changed  
| command\_fontpicker\_changed | command\_html\_link\_clicked |  
command\_kill\_focus | command\_left\_click | command\_left\_dclick  
| command\_list\_begin\_drag | command\_list\_begin\_label\_edit  
| command\_list\_begin\_rdrag | command\_list\_cache\_hint |  
command\_list\_col\_begin\_drag | command\_list\_col\_click |  
command\_list\_col\_dragging | command\_list\_col\_end\_drag |  
command\_list\_col\_right\_click | command\_list\_delete\_all\_items  
| command\_list\_delete\_item | command\_list\_end\_label\_edit  
| command\_list\_insert\_item | command\_list\_item\_activated |  
command\_list\_item\_deselected | command\_list\_item\_focused |  
command\_list\_item\_middle\_click | command\_list\_item\_right\_click  
| command\_list\_item\_selected | command\_list\_key\_down |  
command\_listbox\_doubleclicked | command\_listbox\_selected  
| command\_menu\_selected | command\_notebook\_page\_changed |  
command\_notebook\_page\_changing | command\_radiobox\_selected  
| command\_radiobutton\_selected | command\_right\_click |  
command\_scrollbar\_updated | command\_set\_focus | command\_slider\_updated  
| command\_spincrl\_updated | command\_splitter\_doubleclicked |  
command\_splitter\_sash\_pos\_changed | command\_splitter\_sash\_pos\_changing  
| command\_splitter\_unsplit | command\_text\_enter | command\_text\_updated |  
command\_togglebutton\_clicked | command\_tool\_enter | command\_tool\_rclicked  
| command\_tree\_begin\_drag | command\_tree\_begin\_label\_edit |  
command\_tree\_begin\_rdrag | command\_tree\_delete\_item | command\_tree\_end\_drag  
| command\_tree\_end\_label\_edit | command\_tree\_get\_info |  
command\_tree\_item\_activated | command\_tree\_item\_collapsed |  
command\_tree\_item\_collapsing | command\_tree\_item\_expanded |  
command\_tree\_item\_expanding | command\_tree\_item\_gettooltip  
| command\_tree\_item\_menu | command\_tree\_item\_middle\_click  
| command\_tree\_item\_right\_click | command\_tree\_key\_down |  
command\_tree\_sel\_changed | command\_tree\_sel\_changing | command\_tree\_set\_info  
| command\_tree\_state\_image\_click | command\_vlbox\_selected | context\_menu  
| create | date\_changed | destroy | detailed\_help | display\_changed |  
end\_session | enter\_window | erase\_background | grid\_cell\_begin\_drag  
| grid\_cell\_change | grid\_cell\_left\_click | grid\_cell\_left\_dclick  
| grid\_cell\_right\_click | grid\_cell\_right\_dclick | grid\_col\_size  
| grid\_editor\_created | grid\_editor\_hidden | grid\_editor\_shown |  
grid\_label\_left\_click | grid\_label\_left\_dclick | grid\_label\_right\_click  
| grid\_label\_right\_dclick | grid\_range\_select | grid\_row\_size |  
grid\_select\_cell | help | iconize | idle | joy\_button\_down | joy\_button\_up  
| joy\_move | joy\_zmove | key\_down | key\_up | kill\_focus | leave\_window

```
| left_dclick | left_down | left_up | maximize | menu_close |
menu_highlight | menu_open | middle_dclick | middle_down | middle_up
| motion | mouse_capture_changed | mousewheel | move | navigation_key
| nc_enter_window | nc_leave_window | nc_left_dclick | nc_left_down |
nc_left_up | nc_middle_dclick | nc_middle_down | nc_middle_up | nc_motion
| nc_paint | nc_right_dclick | nc_right_down | nc_right_up | paint |
paint_icon | palette_changed | query_end_session | query_new_palette
| right_dclick | right_down | right_up | sash_dragged | scroll_bottom
| scroll_changed | scroll_linedown | scroll_lineup | scroll_pagedown |
scroll_pageup | scroll_thumbrelease | scroll_thumbtrack | scroll_top |
scrollwin_bottom | scrollwin_linedown | scrollwin_lineup | scrollwin_pagedown
| scrollwin_pageup | scrollwin_thumbrelease | scrollwin_thumbtrack |
scrollwin_top | set_cursor | set_focus | show | size | spin | spin_down
| spin_up | stc_autocomp_selection | stc_calltip_click | stc_change |
stc_charadded | stc_do_drop | stc_doubleclick | stc_drag_over | stc_dwellend
| stc_dwellstart | stc_hotspot_click | stc_hotspot_dclick | stc_key
| stc_macrorecord | stc_marginclick | stc_modified | stc_needshown |
stc_painted | stc_romodifyattempt | stc_savepointleft | stc_savepointreached
| stc_start_drag | stc_styleneeded | stc_updateui | stc_uridropped |
stc_userlistselection | stc_zoom | sys_colour_changed | update_ui
wxEvtHandler()
```

An object reference

```
wxFileDirPicker() = #wxFileDirPicker{type=wxEventType(), path=string()}
```

EventType:

*command\_filepicker\_changed, command\_dirpicker\_changed*

Callback event: *wxFileDirPickerEvent*

```
wxFocus() = #wxFocus{type=wxEventType()}
```

EventType:

*set\_focus, kill\_focus*

Callback event: *wxFocusEvent*

```
wxFontPicker() = #wxFontPicker{type=wxEventType(), font=wxFont() (see module
wxFont)}
```

EventType:

*command\_fontpicker\_changed*

Callback event: *wxFontPickerEvent*

```
wxGrid() = #wxGrid{type=wxEventType(), row=integer(), col=integer(),
x=integer(), y=integer(), selecting=bool(), control=bool(), meta=bool(),
shift=bool(), alt=bool()}
```

EventType:

*grid\_cell\_left\_click, grid\_cell\_right\_click, grid\_cell\_left\_dclick, grid\_cell\_right\_dclick, grid\_label\_left\_click, grid\_label\_right\_click, grid\_label\_left\_dclick, grid\_label\_right\_dclick, grid\_row\_size, grid\_col\_size, grid\_range\_select, grid\_cell\_change, grid\_select\_cell, grid\_editor\_shown, grid\_editor\_hidden, grid\_editor\_created, grid\_cell\_begin\_drag*

Callback event: *wxGridEvent*

```

wxHelp() = #wxHelp{type=wxEventType()}
    EventType:
        help, detailed_help
    Callback event: wxHelpEvent

wxHtmlLink() = #wxHtmlLink{type=wxEventType(), linkInfo=wxHtmlLinkInfo() (see
module wx)}
    EventType:
        command_html_link_clicked
    Callback event: wxHtmlLinkEvent

wxIconize() = #wxIconize{type=wxEventType()}
    EventType:
        iconize
    Callback event: wxIconizeEvent

wxIdle() = #wxIdle{type=wxEventType()}
    EventType:
        idle
    Callback event: wxIdleEvent

wxJoystick() = #wxJoystick{type=wxEventType()}
    EventType:
        joy_button_down, joy_button_up, joy_move, joy_zmove
    Callback event: wxJoystickEvent

wxKey() = #wxKey{type=wxEventType(), x=integer(), y=integer(),
keyCode=integer(), controlDown=bool(), shiftDown=bool(), altDown=bool(),
metaDown=bool(), scanCode=bool(), uniChar=integer(), rawCode=integer(),
rawFlags=integer()}
    EventType:
        char, char_hook, key_down, key_up
    Callback event: wxKeyEvent

wxList() = #wxList{type=wxEventType(), code=integer(),
oldItemIndex=integer(), itemIndex=integer(), col=integer(),
pointDrag={X::integer(), Y::integer()}}
    EventType:
        command_list_begin_drag,          command_list_begin_rdrag,          command_list_begin_label_edit,
        command_list_end_label_edit,      command_list_delete_item,          command_list_delete_all_items,
        command_list_key_down, command_list_insert_item, command_list_col_click, command_list_col_right_click,
        command_list_col_begin_drag,      command_list_col_dragging,        command_list_col_end_drag,
        command_list_item_selected,        command_list_item_deselected,     command_list_item_right_click,
        command_list_item_middle_click,    command_list_item_activated,      command_list_item_focused,
        command_list_cache_hint
    Callback event: wxListEvent

```

`wxMaximize()` = `#wxMaximize{type=wxEventType() }`

EventType:

*maximize*

Callback event: *wxMaximizeEvent*

`wxMenu()` = `#wxMenu{type=wxEventType() }`

EventType:

*menu\_open, menu\_close, menu\_highlight*

Callback event: *wxMenuEvent*

`wxMouse()` = `#wxMouse{type=wxEventType(), x=integer(), y=integer(),  
leftDown=bool(), middleDown=bool(), rightDown=bool(), controlDown=bool(),  
shiftDown=bool(), altDown=bool(), metaDown=bool(), wheelRotation=integer(),  
wheelDelta=integer(), linesPerAction=integer() }`

EventType:

*left\_down, left\_up, middle\_down, middle\_up, right\_down, right\_up, motion, enter\_window, leave\_window,  
left\_dclick, middle\_dclick, right\_dclick, mousewheel, nc\_left\_down, nc\_left\_up, nc\_middle\_down,  
nc\_middle\_up, nc\_right\_down, nc\_right\_up, nc\_motion, nc\_enter\_window, nc\_leave\_window, nc\_left\_dclick,  
nc\_middle\_dclick, nc\_right\_dclick*

Callback event: *wxMouseEvent*

`wxMouseCaptureChanged()` = `#wxMouseCaptureChanged{type=wxEventType() }`

EventType:

*mouse\_capture\_changed*

Callback event: *wxMouseCaptureChangedEvent*

`wxMove()` = `#wxMove{type=wxEventType() }`

EventType:

*move*

Callback event: *wxMoveEvent*

`wxNavigationKey()` = `#wxNavigationKey{type=wxEventType(), flags=integer(),  
focus=wxWindow() (see module wxWindow) }`

EventType:

*navigation\_key*

Callback event: *wxNavigationKeyEvent*

`wxNcPaint()` = `#wxNcPaint{type=wxEventType() }`

EventType:

*nc\_paint*

Callback event: *wxNcPaintEvent*

`wxNotebook()` = `#wxNotebook{type=wxEventType() }`

EventType:

*command\_notebook\_page\_changed, command\_notebook\_page\_changing*

Callback event: *wxNotebookEvent*

```
wxPaint() = #wxPaint{type=wxEventType()}
    EventType:
        paint, paint_icon
    Callback event: wxPaintEvent

wxPaletteChanged() = #wxPaletteChanged{type=wxEventType()}
    EventType:
        palette_changed
    Callback event: wxPaletteChangedEvent

wxQueryNewPalette() = #wxQueryNewPalette{type=wxEventType()}
    EventType:
        query_new_palette
    Callback event: wxQueryNewPaletteEvent

wxSash() = #wxSash{type=wxEventType(), edge=WxSashEdgePosition,
dragRect={X::integer(), Y::integer(), W::integer(), H::integer()},
dragStatus=WxSashDragStatus}
    EventType:
        sash_dragged
    Callback event: wxSashEvent

wxScroll() = #wxScroll{type=wxEventType(), commandInt=integer(),
extraLong=integer()}
    EventType:
        scroll_top, scroll_bottom, scroll_lineup, scroll_linedown, scroll_pageup, scroll_pagedown, scroll_thumbtrack,
        scroll_thumbrelease, scroll_changed
    Callback event: wxScrollEvent

wxScrollWin() = #wxScrollWin{type=wxEventType()}
    EventType:
        scrollwin_top, scrollwin_bottom, scrollwin_lineup, scrollwin_linedown, scrollwin_pageup,
        scrollwin_pagedown, scrollwin_thumbtrack, scrollwin_thumbrelease
    Callback event: wxScrollWinEvent

wxSetCursor() = #wxSetCursor{type=wxEventType()}
    EventType:
        set_cursor
    Callback event: wxSetCursorEvent

wxShow() = #wxShow{type=wxEventType()}
    EventType:
        show
    Callback event: wxShowEvent
```

```
wxSize() = #wxSize{type=wxEventType(), size={W::integer(), H::integer()},  
rect={X::integer(), Y::integer(), W::integer(), H::integer()}}
```

EventType:

*size*

Callback event: *wxSizeEvent*

```
wxSpin() = #wxSpin{type=wxEventType(), commandInt=integer()}
```

EventType:

*command\_spincnt\_updated, spin\_up, spin\_down, spin*

Callback event: *wxSpinEvent*

```
wxSplitter() = #wxSplitter{type=wxEventType()}
```

EventType:

*command\_splitter\_sash\_pos\_changed,* *command\_splitter\_sash\_pos\_changing,*  
*command\_splitter\_doubleclicked, command\_splitter\_unsplit*

Callback event: *wxSplitterEvent*

```
wxStyledText() = #wxStyledText{type=wxEventType(), position=integer(),  
key=integer(), modifiers=integer(), modificationType=integer(),  
text=string(), length=integer(), linesAdded=integer(), line=integer(),  
foldLevelNow=integer(), foldLevelPrev=integer(), margin=integer(),  
message=integer(), wParam=integer(), lParam=integer(), listType=integer(),  
x=integer(), y=integer(), dragText=string(), dragAllowMove=bool(),  
dragResult=WxDragResult}
```

EventType:

*stc\_change, stc\_styleneeded, stc\_charadded, stc\_savepointreached, stc\_savepointleft, stc\_romodifyattempt,*  
*stc\_key, stc\_doubleclick, stc\_updateui, stc\_modified, stc\_macrorecord, stc\_marginclick, stc\_needshown,*  
*stc\_painted, stc\_userlistselection, stc\_uridropped, stc\_dwellstart, stc\_dwellend, stc\_start\_drag, stc\_drag\_over,*  
*stc\_do\_drop, stc\_zoom, stc\_hotspot\_click, stc\_hotspot\_dclick, stc\_calltip\_click, stc\_autocomp\_selection*

Callback event: *wxStyledTextEvent*

```
wxSysColourChanged() = #wxSysColourChanged{type=wxEventType()}
```

EventType:

*sys\_colour\_changed*

Callback event: *wxSysColourChangedEvent*

```
wxTree() = #wxTree{type=wxEventType(), item=integer(), itemOld=integer(),  
pointDrag={X::integer(), Y::integer()}}
```

EventType:

*command\_tree\_begin\_drag,* *command\_tree\_begin\_rdrag,* *command\_tree\_begin\_label\_edit,*  
*command\_tree\_end\_label\_edit, command\_tree\_delete\_item, command\_tree\_get\_info, command\_tree\_set\_info,*  
*command\_tree\_item\_expanded,* *command\_tree\_item\_expanding,* *command\_tree\_item\_collapsed,*  
*command\_tree\_item\_collapsing,* *command\_tree\_sel\_changed,* *command\_tree\_sel\_changing,*  
*command\_tree\_key\_down,* *command\_tree\_item\_activated,* *command\_tree\_item\_right\_click,*  
*command\_tree\_item\_middle\_click,* *command\_tree\_end\_drag,* *command\_tree\_state\_image\_click,*  
*command\_tree\_item\_gettooltip, command\_tree\_item\_menu*

Callback event: *wxTreeEvent*



```
wxUpdateUI() = #wxUpdateUI{type=wxEventType()}
  EventType:
    update_ui
  Callback event: wxUpdateUIEvent

wxWindowCreate() = #wxWindowCreate{type=wxEventType()}
  EventType:
    create
  Callback event: wxWindowCreateEvent

wxWindowDestroy() = #wxWindowDestroy{type=wxEventType()}
  EventType:
    destroy
  Callback event: wxWindowDestroyEvent
```

## Exports

```
connect(This::wxEvtHandler(), EventType::wxEventType()) -> ok
```

Equivalent to *connect(This, EventType, [])*

```
connect(This::wxEvtHandler(), EventType::wxEventType(), Options::[Options]) -> ok
```

This function subscribes the to events of EventType, in the range id, lastId. The events will be received as messages if no callback is supplied.

Options: {id, integer()}, The identifier (or first of the identifier range) to be associated with this event handler. Default ?wxID\_ANY {lastId, integer()}, The second part of the identifier range. If used 'id' must be set as the starting identifier range. Default ?wxID\_ANY {skip, boolean()}, If skip is true further event\_handlers will be called. This is not used if the 'callback' option is used. Default false. {callback, function()} Use a callback fun(EventRecord::wx(), EventObject::wxObject()) to process the event. Default not specified i.e. a message will be delivered to the process calling this function. {userData, term()} An erlang term that will be sent with the event. Default: [].

```
disconnect(This::wxEvtHandler()) -> true | false
```

Equivalent to *disconnect(This, null, [])* Can also have an optional callback Fun() as an additional last argument.

```
disconnect(This::wxEvtHandler(), EventType::wxEventType()) -> true | false
```

Equivalent to *disconnect(This, EventType, [])*

```
disconnect(This::wxEvtHandler(), EventType::wxEventType(), Opts) -> true | false
```

See **external documentation** This function unsubscribes the process or callback fun from the event handler. EventType may be the atom 'null' to match any eventtype. Notice that the options skip and userdata is not used to match the eventhandler.

## wxFileDataObject

---

Erlang module

See external documentation: **wxFileDataObject**.

This class is derived (and can use functions) from:  
*wxDataObject*

### DATA TYPES

`wxFileDataObject()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxFileDataObject()`

See external documentation.

`addFile(This::wxFileDataObject(), Filename::string()) -> ok`

See external documentation.

`getFilenames(This::wxFileDataObject()) -> [[string()]]`

See external documentation.

`destroy(This::wxFileDataObject()) -> ok`

Destroys this object, do not use object again

## wxFileDialog

---

Erlang module

See external documentation: **wxFileDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxFileDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Parent::wxWindow() (see module wxWindow)) -> wxFileDialog()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxFileDialog()`

Types:

`Option = {message, string()} | {defaultDir, string()} | {defaultFile, string()} | {wildCard, string()} | {style, integer()} | {pos, {X::integer(), Y::integer()}} | {sz, {W::integer(), H::integer()}}`

See external documentation.

`getDirectory(This::wxFileDialog()) -> string()`

See external documentation.

`getFilename(This::wxFileDialog()) -> string()`

See external documentation.

`getFilenames(This::wxFileDialog()) -> [[string()]]`

See external documentation.

`getFilterIndex(This::wxFileDialog()) -> integer()`

See external documentation.

`getMessage(This::wxFileDialog()) -> string()`

See external documentation.

`getPath(This::wxFileDialog()) -> string()`

See external documentation.

`getPaths(This::wxFileDialog()) -> [[string()]]`

See external documentation.

`getWildcard(This::wxFileDialog()) -> string()`

See external documentation.

`setDirectory(This::wxFileDialog(), Dir::string()) -> ok`

See external documentation.

`setFilename(This::wxFileDialog(), Name::string()) -> ok`

See external documentation.

`setFilterIndex(This::wxFileDialog(), FilterIndex::integer()) -> ok`

See external documentation.

`setMessage(This::wxFileDialog(), Message::string()) -> ok`

See external documentation.

`setPath(This::wxFileDialog(), Path::string()) -> ok`

See external documentation.

`setWildcard(This::wxFileDialog(), WildCard::string()) -> ok`

See external documentation.

`destroy(This::wxFileDialog()) -> ok`

Destroys this object, do not use object again

## wxFileDirPickerEvent

---

Erlang module

See external documentation: **wxFileDirPickerEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*command\_filepicker\_changed, command\_dirpicker\_changed*

See also the message variant *#wxFileDirPicker{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

### DATA TYPES

`wxFileDirPickerEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

`getPath(This::wxFileDirPickerEvent()) -> string()`

See **external documentation**.

## wxFilePickerCtrl

---

Erlang module

See external documentation: **wxFilePickerCtrl**.

This class is derived (and can use functions) from:

*wxPickerBase*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxFilePickerCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxFilePickerCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxFilePickerCtrl()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxFilePickerCtrl()`

Types:

**Option** = {path, string()} | {message, string()} | {wildcard, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxFilePickerCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, [])`.

`create(This::wxFilePickerCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {path, string()} | {message, string()} | {wildcard, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

**getPath(This::wxFilePickerCtrl()) -> string()**

See external documentation.

**setPath(This::wxFilePickerCtrl(), Str::string()) -> ok**

See external documentation.

**destroy(This::wxFilePickerCtrl()) -> ok**

Destroys this object, do not use object again

## wxFindReplaceData

---

Erlang module

See external documentation: **wxFindReplaceData**.

### DATA TYPES

`wxFindReplaceData()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxFindReplaceData()`

See external documentation.

`new(Flags::integer()) -> wxFindReplaceData()`

See external documentation.

`getFindString(This::wxFindReplaceData()) -> string()`

See external documentation.

`getReplaceString(This::wxFindReplaceData()) -> string()`

See external documentation.

`getFlags(This::wxFindReplaceData()) -> integer()`

See external documentation.

`setFlags(This::wxFindReplaceData(), Flags::integer()) -> ok`

See external documentation.

`setFindString(This::wxFindReplaceData(), Str::string()) -> ok`

See external documentation.

`setReplaceString(This::wxFindReplaceData(), Str::string()) -> ok`

See external documentation.

`destroy(This::wxFindReplaceData()) -> ok`

Destroys this object, do not use object again



## wxFindReplaceDialog

---

Erlang module

See external documentation: **wxFindReplaceDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxFindReplaceDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxFindReplaceDialog()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Data::wxFindReplaceData() (see module wxFindReplaceData), Title::string()) -> wxFindReplaceDialog()`

Equivalent to `new(Parent, Data, Title, [])`.

`new(Parent::wxWindow() (see module wxWindow), Data::wxFindReplaceData() (see module wxFindReplaceData), Title::string(), Options::[Option]) -> wxFindReplaceDialog()`

Types:

**Option** = {style, integer()}

See external documentation.

`create(This::wxFindReplaceDialog(), Parent::wxWindow() (see module wxWindow), Data::wxFindReplaceData() (see module wxFindReplaceData), Title::string()) -> bool()`

Equivalent to `create(This, Parent, Data, Title, [])`.

`create(This::wxFindReplaceDialog(), Parent::wxWindow() (see module wxWindow), Data::wxFindReplaceData() (see module wxFindReplaceData), Title::string(), Options::[Option]) -> bool()`

Types:

**Option** = {style, integer()}

See external documentation.

## **wxFindReplaceDialog**

---

**getData(This::wxFindReplaceDialog()) -> wxFindReplaceData()** (see module **wxFindReplaceData**)

See **external documentation**.

**destroy(This::wxFindReplaceDialog()) -> ok**

Destroys this object, do not use object again

## wxFlexGridSizer

---

Erlang module

See external documentation: **wxFlexGridSizer**.

This class is derived (and can use functions) from:

*wxGridSizer*

*wxSizer*

### DATA TYPES

`wxFlexGridSizer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Cols::integer()) -> wxFlexGridSizer()`

Equivalent to `new(Cols, [])`.

`new(Cols::integer(), Options::[Option]) -> wxFlexGridSizer()`

Types:

**Option** = {vgap, integer()} | {hgap, integer()}

See external documentation.

`new(Rows::integer(), Cols::integer(), Vgap::integer(), Hgap::integer()) -> wxFlexGridSizer()`

See external documentation.

`addGrowbleCol(This::wxFlexGridSizer(), Idx::integer()) -> ok`

Equivalent to `addGrowbleCol(This, Idx, [])`.

`addGrowbleCol(This::wxFlexGridSizer(), Idx::integer(), Options::[Option]) -> ok`

Types:

**Option** = {proportion, integer()}

See external documentation.

`addGrowbleRow(This::wxFlexGridSizer(), Idx::integer()) -> ok`

Equivalent to `addGrowbleRow(This, Idx, [])`.

`addGrowbleRow(This::wxFlexGridSizer(), Idx::integer(), Options::[Option]) -> ok`

Types:

**Option** = {proportion, integer()}

See [external documentation](#).

**getFlexibleDirection(This::wxFlexGridSizer()) -> integer()**

See [external documentation](#).

**getNonFlexibleGrowMode(This::wxFlexGridSizer()) -> WxFlexSizerGrowMode**

Types:

**WxFlexSizerGrowMode = integer()**

See [external documentation](#).

WxFlexSizerGrowMode is one of ?wxFLEX\_GROWMODE\_NONE | ?wxFLEX\_GROWMODE\_SPECIFIED | ?wxFLEX\_GROWMODE\_ALL

**removeGrowableCol(This::wxFlexGridSizer(), Idx::integer()) -> ok**

See [external documentation](#).

**removeGrowableRow(This::wxFlexGridSizer(), Idx::integer()) -> ok**

See [external documentation](#).

**setFlexibleDirection(This::wxFlexGridSizer(), Direction::integer()) -> ok**

See [external documentation](#).

**setNonFlexibleGrowMode(This::wxFlexGridSizer(), Mode::WxFlexSizerGrowMode) -> ok**

Types:

**WxFlexSizerGrowMode = integer()**

See [external documentation](#).

WxFlexSizerGrowMode is one of ?wxFLEX\_GROWMODE\_NONE | ?wxFLEX\_GROWMODE\_SPECIFIED | ?wxFLEX\_GROWMODE\_ALL

**destroy(This::wxFlexGridSizer()) -> ok**

Destroys this object, do not use object again

## wxFocusEvent

---

Erlang module

See external documentation: **wxFocusEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*set\_focus, kill\_focus*

See also the message variant *#wxFocus{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

`wxFocusEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getWindow(This::wxFocusEvent()) -> wxWindow()` (see module `wxWindow`)

See **external documentation**.

## wxFont

---

Erlang module

See external documentation: **wxFont**.

## DATA TYPES

`wxFont()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxFont()`

See external documentation.

`new(Fontname::string()) -> wxFont()`

See external documentation.

`new(Size::integer(), Family::WxFontFamily, Style::WxFontStyle, Weight::integer()) -> wxFont()`

Equivalent to `new(Size, Family, Style, Weight, [])`.

`new(Size::integer(), Family::WxFontFamily, Style::WxFontStyle, Weight::integer(), Options::[Option]) -> wxFont()`

Types:

**Option** = {underlined, bool()} | {face, string()} | {encoding, WxFontEncoding}

**WxFontFamily** = integer()

**WxFontStyle** = integer()

**WxFontEncoding** = integer()

See external documentation.

WxFontFamily is one of ?wxFONTFAMILY\_DEFAULT | ?wxFONTFAMILY\_DECORATIVE | ?wxFONTFAMILY\_ROMAN | ?wxFONTFAMILY\_SCRIPT | ?wxFONTFAMILY\_SWISS | ?wxFONTFAMILY\_MODERN | ?wxFONTFAMILY\_TELETYPE | ?wxFONTFAMILY\_MAX | ?wxFONTFAMILY\_UNKNOWN

WxFontStyle is one of ?wxFONTSTYLE\_NORMAL | ?wxFONTSTYLE\_ITALIC | ?wxFONTSTYLE\_SLANT | ?wxFONTSTYLE\_MAX

WxFontEncoding is one of ?wxFONTENCODING\_SYSTEM | ?wxFONTENCODING\_DEFAULT | ?wxFONTENCODING\_ISO8859\_1 | ?wxFONTENCODING\_ISO8859\_2 | ?wxFONTENCODING\_ISO8859\_3 | ?wxFONTENCODING\_ISO8859\_4 | ?wxFONTENCODING\_ISO8859\_5 | ?wxFONTENCODING\_ISO8859\_6 | ?wxFONTENCODING\_ISO8859\_7 | ?wxFONTENCODING\_ISO8859\_8 | ?wxFONTENCODING\_ISO8859\_9 | ?wxFONTENCODING\_ISO8859\_10 | ?wxFONTENCODING\_ISO8859\_11 | ?wxFONTENCODING\_ISO8859\_12 | ?wxFONTENCODING\_ISO8859\_13 | ?wxFONTENCODING\_ISO8859\_14 | ?wxFONTENCODING\_ISO8859\_15 | ?wxFONTENCODING\_ISO8859\_MAX | ?wxFONTENCODING\_KOI8 | ?wxFONTENCODING\_KOI8\_U | ?wxFONTENCODING\_ALTERNATIVE | ?wxFONTENCODING\_BULGARIAN | ?wxFONTENCODING\_CP437 | ?wxFONTENCODING\_CP850 | ?wxFONTENCODING\_CP852 | ?

wxFONTENCODING_CP855		?wxFONTENCODING_CP866		?wxFONTENCODING_CP874		?	
wxFONTENCODING_CP932		?wxFONTENCODING_CP936		?wxFONTENCODING_CP949		?	
wxFONTENCODING_CP950		?wxFONTENCODING_CP1250		?wxFONTENCODING_CP1251		?	
wxFONTENCODING_CP1252		?wxFONTENCODING_CP1253		?wxFONTENCODING_CP1254		?	
wxFONTENCODING_CP1255		?wxFONTENCODING_CP1256		?wxFONTENCODING_CP1257		?	
wxFONTENCODING_CP12_MAX		?wxFONTENCODING_UTF7		?wxFONTENCODING_UTF8		?	
wxFONTENCODING_EUC_JP		?wxFONTENCODING_UTF16BE		?wxFONTENCODING_UTF16LE		?	
wxFONTENCODING_UTF32BE		?wxFONTENCODING_UTF32LE		?wxFONTENCODING_MACROMAN		?	
	?wxFONTENCODING_MACJAPANESE		?wxFONTENCODING_MACCHINESETRAD		?		
wxFONTENCODING_MACKOREAN		?wxFONTENCODING_MACARABIC		?			
wxFONTENCODING_MACHEBREW		?wxFONTENCODING_MACGREEK		?			
wxFONTENCODING_MACCYRILLIC		?wxFONTENCODING_MACDEVANAGARI		?			
wxFONTENCODING_MACGURMUKHI		?wxFONTENCODING_MACGUJARATI		?			
wxFONTENCODING_MACORIYA		?wxFONTENCODING_MACBENGALI		?			
wxFONTENCODING_MACTAMIL		?wxFONTENCODING_MACTELUGU		?			
wxFONTENCODING_MACKANNADA		?wxFONTENCODING_MACMALAJALAM		?			
wxFONTENCODING_MACSINHALESE		?wxFONTENCODING_MACBURMESE		?			
wxFONTENCODING_MACKHMER		?wxFONTENCODING_MACTHAI		?			
wxFONTENCODING_MACLAOTIAN		?wxFONTENCODING_MACGEORGIAN		?			
wxFONTENCODING_MACARMENIAN		?wxFONTENCODING_MACCHINESESIMP		?			
wxFONTENCODING_MACTIBETAN		?wxFONTENCODING_MACMONGOLIAN		?			
wxFONTENCODING_MACETHIOPIA		?wxFONTENCODING_MACCENTRALEUR		?			
wxFONTENCODING_MACVIATNAMESE		?wxFONTENCODING_MACARABICEXT		?			
wxFONTENCODING_MACSYMBOL		?wxFONTENCODING_MACDINGBATS		?			
wxFONTENCODING_MACTURKISH		?wxFONTENCODING_MACCROATIAN		?			
wxFONTENCODING_MACICELANDIC		?wxFONTENCODING_MACROMANIAN		?			
wxFONTENCODING_MACCELTIC		?wxFONTENCODING_MACGAELIC		?			
wxFONTENCODING_MACKEYBOARD		?wxFONTENCODING_MAX		?wxFONTENCODING_MACMIN		?	
	?wxFONTENCODING_MACMAX		?wxFONTENCODING_UTF16		?wxFONTENCODING_UTF32		?
wxFONTENCODING_UNICODE		?wxFONTENCODING_GB2312		?wxFONTENCODING_BIG5		?	
wxFONTENCODING_SHIFT_JIS							

**isFixedWidth(This::wxFont()) -> bool()**

See [external documentation](#).

**getDefaultEncoding() -> WxFontEncoding**

Types:

**WxFontEncoding = integer()**

See [external documentation](#).

WxFontEncoding is one of ?wxFONTENCODING\_SYSTEM | ?wxFONTENCODING\_DEFAULT | ?wxFONTENCODING\_ISO8859\_1 | ?wxFONTENCODING\_ISO8859\_2 | ?wxFONTENCODING\_ISO8859\_3 | ?wxFONTENCODING\_ISO8859\_4 | ?wxFONTENCODING\_ISO8859\_5 | ?wxFONTENCODING\_ISO8859\_6 | ?wxFONTENCODING\_ISO8859\_7 | ?wxFONTENCODING\_ISO8859\_8 | ?wxFONTENCODING\_ISO8859\_9 | ?wxFONTENCODING\_ISO8859\_10 | ?wxFONTENCODING\_ISO8859\_11 | ?wxFONTENCODING\_ISO8859\_12 | ?wxFONTENCODING\_ISO8859\_13 | ?wxFONTENCODING\_ISO8859\_14 | ?wxFONTENCODING\_ISO8859\_15 | ?wxFONTENCODING\_ISO8859\_MAX | ?wxFONTENCODING\_KOI8 | ?wxFONTENCODING\_KOI8\_U | ?wxFONTENCODING\_ALTERNATIVE | ?wxFONTENCODING\_BULGARIAN | ?wxFONTENCODING\_CP437 | ?wxFONTENCODING\_CP850 | ?wxFONTENCODING\_CP852 | ?wxFONTENCODING\_CP855 | ?wxFONTENCODING\_CP866 | ?wxFONTENCODING\_CP874 | ?wxFONTENCODING\_CP932 | ?wxFONTENCODING\_CP936 | ?wxFONTENCODING\_CP949 | ?

```
wxFONTENCODING_CP950 | ?wxFONTENCODING_CP1250 | ?wxFONTENCODING_CP1251 | ?  
wxFONTENCODING_CP1252 | ?wxFONTENCODING_CP1253 | ?wxFONTENCODING_CP1254 | ?  
wxFONTENCODING_CP1255 | ?wxFONTENCODING_CP1256 | ?wxFONTENCODING_CP1257 | ?  
wxFONTENCODING_CP12_MAX | ?wxFONTENCODING_UTF7 | ?wxFONTENCODING_UTF8 | ?  
wxFONTENCODING_EUC_JP | ?wxFONTENCODING_UTF16BE | ?wxFONTENCODING_UTF16LE | ?  
wxFONTENCODING_UTF32BE | ?wxFONTENCODING_UTF32LE | ?wxFONTENCODING_MACROMAN  
| ?wxFONTENCODING_MACJAPANESE | ?wxFONTENCODING_MACCHINESETRAD | ?  
wxFONTENCODING_MACKOREAN | ?wxFONTENCODING_MACARABIC | ?  
wxFONTENCODING_MACHEBREW | ?wxFONTENCODING_MACGREEK | ?  
wxFONTENCODING_MACCYRILLIC | ?wxFONTENCODING_MACDEVANAGARI | ?  
wxFONTENCODING_MACGURMUKHI | ?wxFONTENCODING_MACGUJARATI | ?  
wxFONTENCODING_MACORIYA | ?wxFONTENCODING_MACBENGALI | ?  
wxFONTENCODING_MACTAMIL | ?wxFONTENCODING_MACTELUGU | ?  
wxFONTENCODING_MACKANNADA | ?wxFONTENCODING_MACMALAJALAM | ?  
wxFONTENCODING_MACSINHALESE | ?wxFONTENCODING_MACBURMESE | ?  
wxFONTENCODING_MACKHMER | ?wxFONTENCODING_MACTHAI | ?  
wxFONTENCODING_MACLAOTIAN | ?wxFONTENCODING_MACGEORGIAN | ?  
wxFONTENCODING_MACARMENIAN | ?wxFONTENCODING_MACCHINESESIMP | ?  
wxFONTENCODING_MACTIBETAN | ?wxFONTENCODING_MACMONGOLIAN | ?  
wxFONTENCODING_MACETHIOPIA | ?wxFONTENCODING_MACCENTRALEUR | ?  
wxFONTENCODING_MACVIATNAMESE | ?wxFONTENCODING_MACARABICEXT | ?  
wxFONTENCODING_MACSYMBOL | ?wxFONTENCODING_MACDINGBATS | ?  
wxFONTENCODING_MACTURKISH | ?wxFONTENCODING_MACCROATIAN | ?  
wxFONTENCODING_MACICELANDIC | ?wxFONTENCODING_MACROMANIAN | ?  
wxFONTENCODING_MACCELTEIC | ?wxFONTENCODING_MACGAELIC | ?  
wxFONTENCODING_MACKEYBOARD | ?wxFONTENCODING_MAX | ?wxFONTENCODING_MACMIN  
| ?wxFONTENCODING_MACMAX | ?wxFONTENCODING_UTF16 | ?wxFONTENCODING_UTF32 | ?  
wxFONTENCODING_UNICODE | ?wxFONTENCODING_GB2312 | ?wxFONTENCODING_BIG5 | ?  
wxFONTENCODING_SHIFT_JIS
```

**getFaceName(This::wxFont()) -> string()**

See [external documentation](#).

**getFamily(This::wxFont()) -> WxFontFamily**

Types:

**WxFontFamily = integer()**

See [external documentation](#).

```
WxFontFamily is one of ?wxFONTFAMILY_DEFAULT | ?wxFONTFAMILY_DECORATIVE  
| ?wxFONTFAMILY_ROMAN | ?wxFONTFAMILY_SCRIPT | ?wxFONTFAMILY_SWISS | ?  
wxFONTFAMILY_MODERN | ?wxFONTFAMILY_TELETYPE | ?wxFONTFAMILY_MAX | ?  
wxFONTFAMILY_UNKNOWN
```

**getNativeFontInfoDesc(This::wxFont()) -> string()**

See [external documentation](#).

**getNativeFontInfoUserDesc(This::wxFont()) -> string()**

See [external documentation](#).



**getPointSize(This::wxFont()) -> integer()**

See external documentation.

**getStyle(This::wxFont()) -> WxFontStyle**

Types:

**WxFontStyle = integer()**

See external documentation.

WxFontStyle is one of ?wxFONTSTYLE\_NORMAL | ?wxFONTSTYLE\_ITALIC | ?wxFONTSTYLE\_SLANT | ?wxFONTSTYLE\_MAX

**getUnderlined(This::wxFont()) -> bool()**

See external documentation.

**getWeight(This::wxFont()) -> integer()**

See external documentation.

**ok(This::wxFont()) -> bool()**

See external documentation.

**setDefaultEncoding(Encoding::WxFontEncoding) -> ok**

Types:

**WxFontEncoding = integer()**

See external documentation.

WxFontEncoding is one of ?wxFONTENCODING\_SYSTEM | ?wxFONTENCODING\_DEFAULT | ?wxFONTENCODING\_ISO8859\_1 | ?wxFONTENCODING\_ISO8859\_2 | ?wxFONTENCODING\_ISO8859\_3 | ?wxFONTENCODING\_ISO8859\_4 | ?wxFONTENCODING\_ISO8859\_5 | ?wxFONTENCODING\_ISO8859\_6 | ?wxFONTENCODING\_ISO8859\_7 | ?wxFONTENCODING\_ISO8859\_8 | ?wxFONTENCODING\_ISO8859\_9 | ?wxFONTENCODING\_ISO8859\_10 | ?wxFONTENCODING\_ISO8859\_11 | ?wxFONTENCODING\_ISO8859\_12 | ?wxFONTENCODING\_ISO8859\_13 | ?wxFONTENCODING\_ISO8859\_14 | ?wxFONTENCODING\_ISO8859\_15 | ?wxFONTENCODING\_ISO8859\_MAX | ?wxFONTENCODING\_KOI8 | ?wxFONTENCODING\_KOI8\_U | ?wxFONTENCODING\_ALTERNATIVE | ?wxFONTENCODING\_BULGARIAN | ?wxFONTENCODING\_CP437 | ?wxFONTENCODING\_CP850 | ?wxFONTENCODING\_CP852 | ?wxFONTENCODING\_CP855 | ?wxFONTENCODING\_CP866 | ?wxFONTENCODING\_CP874 | ?wxFONTENCODING\_CP932 | ?wxFONTENCODING\_CP936 | ?wxFONTENCODING\_CP949 | ?wxFONTENCODING\_CP950 | ?wxFONTENCODING\_CP1250 | ?wxFONTENCODING\_CP1251 | ?wxFONTENCODING\_CP1252 | ?wxFONTENCODING\_CP1253 | ?wxFONTENCODING\_CP1254 | ?wxFONTENCODING\_CP1255 | ?wxFONTENCODING\_CP1256 | ?wxFONTENCODING\_CP1257 | ?wxFONTENCODING\_CP12\_MAX | ?wxFONTENCODING\_UTF7 | ?wxFONTENCODING\_UTF8 | ?wxFONTENCODING\_EUC\_JP | ?wxFONTENCODING\_UTF16BE | ?wxFONTENCODING\_UTF16LE | ?wxFONTENCODING\_UTF32BE | ?wxFONTENCODING\_UTF32LE | ?wxFONTENCODING\_MACROMAN | ?wxFONTENCODING\_MACJAPANESE | ?wxFONTENCODING\_MACCHINESETRAD | ?wxFONTENCODING\_MACKOREAN | ?wxFONTENCODING\_MACARABIC | ?wxFONTENCODING\_MACHEBREW | ?wxFONTENCODING\_MACGREEK | ?wxFONTENCODING\_MACCYRILLIC | ?wxFONTENCODING\_MACDEVANAGARI | ?wxFONTENCODING\_MACGURMUKHI | ?wxFONTENCODING\_MACGUJARATI | ?wxFONTENCODING\_MACORIYA | ?wxFONTENCODING\_MACBENGALI | ?wxFONTENCODING\_MACTAMIL | ?wxFONTENCODING\_MACTELUGU

wxFONTENCODING_MACKANNADA		?wxFONTENCODING_MACMALAJALAM		?
wxFONTENCODING_MACSINHALESE		?wxFONTENCODING_MACBURMESE		?
wxFONTENCODING_MACKHMER		?wxFONTENCODING_MACTHAI		?
wxFONTENCODING_MACLAOTIAN		?wxFONTENCODING_MACGEORGIAN		?
wxFONTENCODING_MACARMENIAN		?wxFONTENCODING_MACCHINESESIMP		?
wxFONTENCODING_MACTIBETAN		?wxFONTENCODING_MACMONGOLIAN		?
wxFONTENCODING_MACETHIOPIC		?wxFONTENCODING_MACCENTRALEUR		?
wxFONTENCODING_MACVIATNAMESE		?wxFONTENCODING_MACARABICEXT		?
wxFONTENCODING_MACSYMBOL		?wxFONTENCODING_MACDINGBATS		?
wxFONTENCODING_MACTURKISH		?wxFONTENCODING_MACCROATIAN		?
wxFONTENCODING_MACICELANDIC		?wxFONTENCODING_MACROMANIAN		?
wxFONTENCODING_MACCELTIC		?wxFONTENCODING_MACGAELIC		?
wxFONTENCODING_MACKEYBOARD		?wxFONTENCODING_MAX		?wxFONTENCODING_MACMIN
?wxFONTENCODING_MACMAX		?wxFONTENCODING_UTF16		?wxFONTENCODING_UTF32
?wxFONTENCODING_UNICODE		?wxFONTENCODING_GB2312		?wxFONTENCODING_BIG5
?wxFONTENCODING_SHIFT_JIS				

**setFaceName(This::wxFont(), FaceName::string()) -> bool()**

See [external documentation](#).

**setFamily(This::wxFont(), Family::WxFontFamily) -> ok**

Types:

**WxFontFamily = integer()**

See [external documentation](#).

WxFontFamily is one of ?wxFONTFAMILY\_DEFAULT | ?wxFONTFAMILY\_DECORATIVE  
| ?wxFONTFAMILY\_ROMAN | ?wxFONTFAMILY\_SCRIPT | ?wxFONTFAMILY\_SWISS | ?  
wxFONTFAMILY\_MODERN | ?wxFONTFAMILY\_TELETYPE | ?wxFONTFAMILY\_MAX | ?  
wxFONTFAMILY\_UNKNOWN

**setPointSize(This::wxFont(), PointSize::integer()) -> ok**

See [external documentation](#).

**setStyle(This::wxFont(), Style::WxFontStyle) -> ok**

Types:

**WxFontStyle = integer()**

See [external documentation](#).

WxFontStyle is one of ?wxFONTSTYLE\_NORMAL | ?wxFONTSTYLE\_ITALIC | ?wxFONTSTYLE\_SLANT | ?  
wxFONTSTYLE\_MAX

**setUnderlined(This::wxFont(), Underlined::bool()) -> ok**

See [external documentation](#).

**setWeight(This::wxFont(), Weight::integer()) -> ok**

See [external documentation](#).

```
destroy(This::wxFont()) -> ok
```

Destroys this object, do not use object again

## wxFontData

---

Erlang module

See external documentation: **wxFontData**.

### DATA TYPES

`wxFontData()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxFontData()`

See external documentation.

`new(Data::wxFontData()) -> wxFontData()`

See external documentation.

`enableEffects(This::wxFontData(), Flag::bool()) -> ok`

See external documentation.

`getAllowSymbols(This::wxFontData()) -> bool()`

See external documentation.

`getColour(This::wxFontData()) -> colour() (see module wx)`

See external documentation.

`getChosenFont(This::wxFontData()) -> wxFont() (see module wxFont)`

See external documentation.

`getEnableEffects(This::wxFontData()) -> bool()`

See external documentation.

`getInitialFont(This::wxFontData()) -> wxFont() (see module wxFont)`

See external documentation.

`getShowHelp(This::wxFontData()) -> bool()`

See external documentation.

`setAllowSymbols(This::wxFontData(), Flag::bool()) -> ok`

See external documentation.

**setChosenFont(This::wxFontData(), Font::wxFont() (see module wxFont)) -> ok**

See external documentation.

**setColour(This::wxFontData(), Colour::colour() (see module wx)) -> ok**

See external documentation.

**setInitialFont(This::wxFontData(), Font::wxFont() (see module wxFont)) -> ok**

See external documentation.

**setRange(This::wxFontData(), MinRange::integer(), MaxRange::integer()) -> ok**

See external documentation.

**setShowHelp(This::wxFontData(), Flag::bool()) -> ok**

See external documentation.

**destroy(This::wxFontData()) -> ok**

Destroys this object, do not use object again

## wxFontDialog

---

Erlang module

See external documentation: **wxFontDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxFontDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxFontDialog()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Data::wxFontData() (see module wxFontData)) -> wxFontDialog()`

See external documentation.

`create(This::wxFontDialog(), Parent::wxWindow() (see module wxWindow), Data::wxFontData() (see module wxFontData)) -> bool()`

See external documentation.

`getFontData(This::wxFontDialog()) -> wxFontData() (see module wxFontData)`

See external documentation.

`destroy(This::wxFontDialog()) -> ok`

Destroys this object, do not use object again

# wxFontPickerCtrl

Erlang module

See external documentation: **wxFontPickerCtrl**.

This class is derived (and can use functions) from:

*wxPickerBase*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxFontPickerCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxFontPickerCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxFontPickerCtrl()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxFontPickerCtrl()`

Types:

**Option** = {initial, wxFont() (see module wxFont)} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxFontPickerCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, [])`.

`create(This::wxFontPickerCtrl(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {initial, wxFont() (see module wxFont)} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`getSelectedFont(This::wxFontPickerCtrl()) -> wxFont() (see module wxFont)`

See [external documentation](#).

`setSelectedFont(This::wxFontPickerCtrl(), F::wxFont() (see module wxFont)) -> ok`

See [external documentation](#).

`getMaxPointSize(This::wxFontPickerCtrl()) -> integer()`

See [external documentation](#).

`setMaxPointSize(This::wxFontPickerCtrl(), Max::integer()) -> ok`

See [external documentation](#).

`destroy(This::wxFontPickerCtrl()) -> ok`

Destroys this object, do not use object again



## wxFontPickerEvent

---

Erlang module

See external documentation: **wxFontPickerEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*command\_fontpicker\_changed*

See also the message variant *#wxFontPicker{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

### DATA TYPES

`wxFontPickerEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

`getFont(This::wxFontPickerEvent()) -> wxFont()` (see module `wxFont`)

See **external documentation**.

## wxFrame

---

Erlang module

See external documentation: **wxFrame**.

This class is derived (and can use functions) from:

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxFrame()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxFrame()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string())  
-> wxFrame()`

Equivalent to `new(Parent, Id, Title, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string(),  
Options::[Option]) -> wxFrame()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxFrame(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Title::string()) -> bool()`

Equivalent to `create(This, Parent, Id, Title, [])`.

`create(This::wxFrame(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Title::string(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`createStatusBar(This::wxFrame()) -> wxStatusBar() (see module wxStatusBar)`

Equivalent to `createStatusBar(This, [])`.

`createStatusBar(This::wxFrame(), Options::[Option]) -> wxStatusBar()` (see module `wxStatusBar`)

Types:

`Option = {number, integer()} | {style, integer()} | {id, integer()}`

See external documentation.

`createToolBar(This::wxFrame()) -> wxToolBar()` (see module `wxToolBar`)

Equivalent to `createToolBar(This, [])`.

`createToolBar(This::wxFrame(), Options::[Option]) -> wxToolBar()` (see module `wxToolBar`)

Types:

`Option = {style, integer()} | {id, integer()}`

See external documentation.

`getClientAreaOrigin(This::wxFrame()) -> {X::integer(), Y::integer()}`

See external documentation.

`getMenuBar(This::wxFrame()) -> wxMenuBar()` (see module `wxMenuBar`)

See external documentation.

`getStatusBar(This::wxFrame()) -> wxStatusBar()` (see module `wxStatusBar`)

See external documentation.

`getStatusBarPane(This::wxFrame()) -> integer()`

See external documentation.

`getToolBar(This::wxFrame()) -> wxToolBar()` (see module `wxToolBar`)

See external documentation.

`processCommand(This::wxFrame(), Winid::integer()) -> bool()`

See external documentation.

`sendSizeEvent(This::wxFrame()) -> ok`

See external documentation.

`setMenuBar(This::wxFrame(), Menubar::wxMenuBar())` (see module `wxMenuBar`) -> `ok`

See external documentation.

`setStatusBar(This::wxFrame(), Statbar::wxStatusBar())` (see module `wxStatusBar`) -> `ok`

See external documentation.

**setStatusBarPane(This::wxFrame(), N::integer()) -> ok**

See [external documentation](#).

**setStatusText(This::wxFrame(), Text::string()) -> ok**

Equivalent to *setStatusText(This, Text, [])*.

**setStatusText(This::wxFrame(), Text::string(), Options::[Option]) -> ok**

Types:

**Option = {number, integer()}**

See [external documentation](#).

**setStatusWidths(This::wxFrame(), Widths\_field::[integer()]) -> ok**

See [external documentation](#).

**setToolBar(This::wxFrame(), Toolbar::wxToolBar() (see module wxToolBar)) -> ok**

See [external documentation](#).

**destroy(This::wxFrame()) -> ok**

Destroys this object, do not use object again

## wxGBSizerItem

---

Erlang module

See external documentation: **wxGBSizerItem**.

This class is derived (and can use functions) from:  
*wxSizerItem*

### DATA TYPES

`wxGBSizerItem()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxGLCanvas

---

Erlang module

See external documentation: **wxGLCanvas**.

This class is derived (and can use functions) from:

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxGLCanvas()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new(Parent::wxWindow() (see module wxWindow)) -> wxGLCanvas()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), X::term()) -> wxGLCanvas()`

See **external documentation**.

Alternatives:

```
new(Parent::wxWindow:wxWindow(), Shared::wxGLContext:wxGLContext() |  
wxGLCanvas()) -> new(Parent, Shared, [])
```

```
new(Parent::wxWindow:wxWindow(), [Option]) -> wxGLCanvas()
```

Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {name, string()} | {attribList, [integer()]} | {palette, wxPalette:wxPalette() }

`new(Parent::wxWindow() (see module wxWindow), Shared::wxGLContext() (see module wxGLContext) | wxGLCanvas(), Options::[Option]) -> wxGLCanvas()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {name, string()} | {attribList, [integer()]} | {palette, wxPalette() (see module wxPalette)}

See **external documentation**.

`getContext(This::wxGLCanvas()) -> wxGLContext() (see module wxGLContext)`

See **external documentation**.

`setCurrent(This::wxGLCanvas()) -> ok`

See **external documentation**.

`swapBuffers(This::wxGLCanvas()) -> ok`

See **external documentation**.

**destroy(This::wxGLCanvas()) -> ok**

Destroys this object, do not use object again

## wxGauge

---

Erlang module

See external documentation: **wxGauge**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxGauge()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxGauge()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Range::integer()) -> wxGauge()`

Equivalent to `new(Parent, Id, Range, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Range::integer(), Options::[Option]) -> wxGauge()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxGauge(), Parent::wxWindow() (see module wxWindow), Id::integer(), Range::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, Range, [])`.

`create(This::wxGauge(), Parent::wxWindow() (see module wxWindow), Id::integer(), Range::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`getBezelFace(This::wxGauge()) -> integer()`

See external documentation.



**getRange(This::wxGauge()) -> integer()**

See external documentation.

**getShadowWidth(This::wxGauge()) -> integer()**

See external documentation.

**getValue(This::wxGauge()) -> integer()**

See external documentation.

**isVertical(This::wxGauge()) -> bool()**

See external documentation.

**setBezelFace(This::wxGauge(), W::integer()) -> ok**

See external documentation.

**setRange(This::wxGauge(), R::integer()) -> ok**

See external documentation.

**setShadowWidth(This::wxGauge(), W::integer()) -> ok**

See external documentation.

**setValue(This::wxGauge(), Pos::integer()) -> ok**

See external documentation.

**pulse(This::wxGauge()) -> ok**

See external documentation.

**destroy(This::wxGauge()) -> ok**

Destroys this object, do not use object again

## wxGenericDirCtrl

---

Erlang module

See external documentation: **wxGenericDirCtrl**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxGenericDirCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxGenericDirCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxGenericDirCtrl()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxGenericDirCtrl()`

Types:

**Option** = {id, integer()} | {dir, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {filter, string()} | {defaultFilter, integer()}

See external documentation.

`create(This::wxGenericDirCtrl(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

`create(This::wxGenericDirCtrl(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {id, integer()} | {dir, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {filter, string()} | {defaultFilter, integer()}

See external documentation.

`init(This::wxGenericDirCtrl()) -> ok`

See external documentation.

`collapseTree(This::wxGenericDirCtrl()) -> ok`

See external documentation.

`expandPath(This::wxGenericDirCtrl(), Path::string()) -> bool()`

See external documentation.

`getDefaultPath(This::wxGenericDirCtrl()) -> string()`

See external documentation.

`getPath(This::wxGenericDirCtrl()) -> string()`

See external documentation.

`getFilePath(This::wxGenericDirCtrl()) -> string()`

See external documentation.

`getFilter(This::wxGenericDirCtrl()) -> string()`

See external documentation.

`getFilterIndex(This::wxGenericDirCtrl()) -> integer()`

See external documentation.

`getRootId(This::wxGenericDirCtrl()) -> integer()`

See external documentation.

`getTreeCtrl(This::wxGenericDirCtrl()) -> wxTreeCtrl() (see module wxTreeCtrl)`

See external documentation.

`reCreateTree(This::wxGenericDirCtrl()) -> ok`

See external documentation.

`setDefaultPath(This::wxGenericDirCtrl(), Path::string()) -> ok`

See external documentation.

`setFilter(This::wxGenericDirCtrl(), Filter::string()) -> ok`

See external documentation.

`setFilterIndex(This::wxGenericDirCtrl(), N::integer()) -> ok`

See external documentation.

`setPath(This::wxGenericDirCtrl(), Path::string()) -> ok`

See external documentation.

```
destroy(This::wxGenericDirCtrl()) -> ok
```

Destroys this object, do not use object again

## wxGraphicsBrush

---

Erlang module

See external documentation: **wxGraphicsBrush**.

This class is derived (and can use functions) from:  
*wxGraphicsObject*

### DATA TYPES

`wxGraphicsBrush( )`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxGraphicsContext

---

Erlang module

See external documentation: **wxGraphicsContext**.

This class is derived (and can use functions) from:  
*wxGraphicsObject*

### DATA TYPES

`wxGraphicsContext()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`create() -> wxGraphicsContext()`

See external documentation.

`create(Dc::wxWindowDC() (see module wxWindowDC) | wxWindow() (see module wxWindow)) -> wxGraphicsContext()`

See external documentation.

`createPen(This::wxGraphicsContext(), Pen::wxPen() (see module wxPen)) -> wxGraphicsPen() (see module wxGraphicsPen)`

See external documentation.

`createBrush(This::wxGraphicsContext(), Brush::wxBrush() (see module wxBrush)) -> wxGraphicsBrush() (see module wxGraphicsBrush)`

See external documentation.

`createRadialGradientBrush(This::wxGraphicsContext(), Xo::float(), Yo::float(), Xc::float(), Yc::float(), Radius::float(), OColor::colour() (see module wx), CColor::colour() (see module wx)) -> wxGraphicsBrush() (see module wxGraphicsBrush)`

See external documentation.

`createLinearGradientBrush(This::wxGraphicsContext(), X1::float(), Y1::float(), X2::float(), Y2::float(), C1::colour() (see module wx), C2::colour() (see module wx)) -> wxGraphicsBrush() (see module wxGraphicsBrush)`

See external documentation.

`createFont(This::wxGraphicsContext(), Font::wxFont() (see module wxFont)) -> wxGraphicsFont() (see module wxGraphicsFont)`

Equivalent to `createFont(This, Font, [])`.

`createFont(This::wxGraphicsContext(), Font::wxFont() (see module wxFont), Options::[Option]) -> wxGraphicsFont() (see module wxGraphicsFont)`

Types:

`Option = {col, colour() (see module wx)}`

See external documentation.

`createMatrix(This::wxGraphicsContext()) -> wxGraphicsMatrix() (see module wxGraphicsMatrix)`

Equivalent to `createMatrix(This, [])`.

`createMatrix(This::wxGraphicsContext(), Options::[Option]) -> wxGraphicsMatrix() (see module wxGraphicsMatrix)`

Types:

`Option = {a, float()} | {b, float()} | {c, float()} | {d, float()} | {tx, float()} | {ty, float()}`

See external documentation.

`createPath(This::wxGraphicsContext()) -> wxGraphicsPath() (see module wxGraphicsPath)`

See external documentation.

`clip(This::wxGraphicsContext(), Region::wxRegion() (see module wxRegion)) -> ok`

See external documentation.

`clip(This::wxGraphicsContext(), X::float(), Y::float(), W::float(), H::float()) -> ok`

See external documentation.

`resetClip(This::wxGraphicsContext()) -> ok`

See external documentation.

`drawBitmap(This::wxGraphicsContext(), Bmp::wxBitmap() (see module wxBitmap), X::float(), Y::float(), W::float(), H::float()) -> ok`

See external documentation.

`drawEllipse(This::wxGraphicsContext(), X::float(), Y::float(), W::float(), H::float()) -> ok`

See external documentation.

`drawIcon(This::wxGraphicsContext(), Icon::wxIcon() (see module wxIcon), X::float(), Y::float(), W::float(), H::float()) -> ok`

See external documentation.

```
drawLines(This::wxGraphicsContext(), N::integer(), Points::{X::float(),  
Y::float()}) -> ok
```

Equivalent to *drawLines(This, N, Points, [])*.

```
drawLines(This::wxGraphicsContext(), N::integer(), Points::{X::float(),  
Y::float()}, Options::[Option]) -> ok
```

Types:

**Option** = {fillStyle, integer()}

See [external documentation](#).

```
drawPath(This::wxGraphicsContext(), Path::wxGraphicsPath() (see module  
wxGraphicsPath)) -> ok
```

Equivalent to *drawPath(This, Path, [])*.

```
drawPath(This::wxGraphicsContext(), Path::wxGraphicsPath() (see module  
wxGraphicsPath), Options::[Option]) -> ok
```

Types:

**Option** = {fillStyle, integer()}

See [external documentation](#).

```
drawRectangle(This::wxGraphicsContext(), X::float(), Y::float(), W::float(),  
H::float()) -> ok
```

See [external documentation](#).

```
drawRoundedRectangle(This::wxGraphicsContext(), X::float(), Y::float(),  
W::float(), H::float(), Radius::float()) -> ok
```

See [external documentation](#).

```
drawText(This::wxGraphicsContext(), Str::string(), X::float(), Y::float()) ->  
ok
```

See [external documentation](#).

```
drawText(This::wxGraphicsContext(), Str::string(), X::float(), Y::float(),  
X::float() | term()) -> ok
```

See [external documentation](#).

Alternatives:

```
drawText(This::wxGraphicsContext(), Str::string(), X::float(), Y::float(),  
Angle::float()) -> ok
```

```
drawText(This::wxGraphicsContext(), Str::string(), X::float(), Y::float(),  
BackgroundBrush::wxGraphicsBrush:wxGraphicsBrush()) -> ok
```



`drawText(This::wxGraphicsContext(), Str::string(), X::float(), Y::float(), Angle::float(), BackgroundBrush::wxGraphicsBrush() (see module wxGraphicsBrush)) -> ok`

See external documentation.

`fillPath(This::wxGraphicsContext(), Path::wxGraphicsPath() (see module wxGraphicsPath)) -> ok`

Equivalent to `fillPath(This, Path, [])`.

`fillPath(This::wxGraphicsContext(), Path::wxGraphicsPath() (see module wxGraphicsPath), Options::[Option]) -> ok`

Types:

`Option = {fillStyle, integer()}`

See external documentation.

`strokePath(This::wxGraphicsContext(), Path::wxGraphicsPath() (see module wxGraphicsPath)) -> ok`

See external documentation.

`getNativeContext(This::wxGraphicsContext()) -> ok`

See external documentation.

`getPartialTextExtents(This::wxGraphicsContext(), Text::string(), Widths::[float()]) -> ok`

See external documentation.

`getTextExtent(This::wxGraphicsContext(), Text::string()) -> {Width::float(), Height::float(), Descent::float(), ExternalLeading::float()}`

See external documentation.

`rotate(This::wxGraphicsContext(), Angle::float()) -> ok`

See external documentation.

`scale(This::wxGraphicsContext(), XScale::float(), YScale::float()) -> ok`

See external documentation.

`translate(This::wxGraphicsContext(), Dx::float(), Dy::float()) -> ok`

See external documentation.

`getTransform(This::wxGraphicsContext()) -> wxGraphicsMatrix() (see module wxGraphicsMatrix)`

See external documentation.

`setTransform(This::wxGraphicsContext(), Matrix::wxGraphicsMatrix() (see module wxGraphicsMatrix)) -> ok`

See external documentation.

`concatTransform(This::wxGraphicsContext(), Matrix::wxGraphicsMatrix() (see module wxGraphicsMatrix)) -> ok`

See external documentation.

`setBrush(This::wxGraphicsContext(), Brush::wxGraphicsBrush() (see module wxGraphicsBrush) | wxBrush() (see module wxBrush)) -> ok`

See external documentation.

`setFont(This::wxGraphicsContext(), Font::wxGraphicsFont() (see module wxGraphicsFont)) -> ok`

See external documentation.

`setFont(This::wxGraphicsContext(), Font::wxFont() (see module wxFont), Colour::colour() (see module wx)) -> ok`

See external documentation.

`setPen(This::wxGraphicsContext(), Pen::wxPen() (see module wxPen) | wxGraphicsPen() (see module wxGraphicsPen)) -> ok`

See external documentation.

`strokeLine(This::wxGraphicsContext(), X1::float(), Y1::float(), X2::float(), Y2::float()) -> ok`

See external documentation.

`strokeLines(This::wxGraphicsContext(), N::integer(), Points::{X::float(), Y::float()}) -> ok`

See external documentation.

`strokeLines(This::wxGraphicsContext(), N::integer(), BeginPoints::{X::float(), Y::float()}, EndPoints::{X::float(), Y::float()}) -> ok`

See external documentation.

## wxGraphicsFont

---

Erlang module

See external documentation: **wxGraphicsFont**.

This class is derived (and can use functions) from:  
*wxGraphicsObject*

### DATA TYPES

`wxGraphicsFont()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxGraphicsMatrix

---

Erlang module

See external documentation: **wxGraphicsMatrix**.

This class is derived (and can use functions) from:  
*wxGraphicsObject*

### DATA TYPES

`wxGraphicsMatrix()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`concat(This::wxGraphicsMatrix(), T::wxGraphicsMatrix()) -> ok`

See external documentation.

`get(This::wxGraphicsMatrix()) -> {A::float(), B::float(), C::float(),  
D::float(), Tx::float(), Ty::float()}`

See external documentation.

`getNativeMatrix(This::wxGraphicsMatrix()) -> ok`

See external documentation.

`invert(This::wxGraphicsMatrix()) -> ok`

See external documentation.

`isEqual(This::wxGraphicsMatrix(), T::wxGraphicsMatrix()) -> bool()`

See external documentation.

`isIdentity(This::wxGraphicsMatrix()) -> bool()`

See external documentation.

`rotate(This::wxGraphicsMatrix(), Angle::float()) -> ok`

See external documentation.

`scale(This::wxGraphicsMatrix(), XScale::float(), YScale::float()) -> ok`

See external documentation.

`translate(This::wxGraphicsMatrix(), Dx::float(), Dy::float()) -> ok`

See external documentation.

```
set(This::wxGraphicsMatrix()) -> ok
```

Equivalent to *set(This, [])*.

```
set(This::wxGraphicsMatrix(), Options::[Option]) -> ok
```

Types:

**Option** = {a, float()} | {b, float()} | {c, float()} | {d, float()} | {tx, float()} | {ty, float()}

See **external documentation**.

```
transformPoint(This::wxGraphicsMatrix()) -> {X::float(), Y::float()}
```

See **external documentation**.

```
transformDistance(This::wxGraphicsMatrix()) -> {Dx::float(), Dy::float()}
```

See **external documentation**.

## wxGraphicsObject

---

Erlang module

See external documentation: **wxGraphicsObject**.

### DATA TYPES

`wxGraphicsObject()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

`getRenderer(This::wxGraphicsObject()) -> wxGraphicsRenderer()` (see module **wxGraphicsRenderer**)

See external documentation.

`isNull(This::wxGraphicsObject()) -> bool()`

See external documentation.

# wxGraphicsPath

---

Erlang module

See external documentation: **wxGraphicsPath**.

This class is derived (and can use functions) from:  
*wxGraphicsObject*

## DATA TYPES

`wxGraphicsPath()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`moveToPoint(This::wxGraphicsPath(), P::{X::float(), Y::float()}) -> ok`

See external documentation.

`moveToPoint(This::wxGraphicsPath(), X::float(), Y::float()) -> ok`

See external documentation.

`addArc(This::wxGraphicsPath(), C::{X::float(), Y::float()}, R::float(),  
StartAngle::float(), EndAngle::float(), Clockwise::bool()) -> ok`

See external documentation.

`addArc(This::wxGraphicsPath(), X::float(), Y::float(), R::float(),  
StartAngle::float(), EndAngle::float(), Clockwise::bool()) -> ok`

See external documentation.

`addArcToPoint(This::wxGraphicsPath(), X1::float(), Y1::float(), X2::float(),  
Y2::float(), R::float()) -> ok`

See external documentation.

`addCircle(This::wxGraphicsPath(), X::float(), Y::float(), R::float()) -> ok`

See external documentation.

`addCurveToPoint(This::wxGraphicsPath(), C1::{X::float(), Y::float()}, C2::  
{X::float(), Y::float()}, E::{X::float(), Y::float()}) -> ok`

See external documentation.

`addCurveToPoint(This::wxGraphicsPath(), Cx1::float(), Cy1::float(),  
Cx2::float(), Cy2::float(), X::float(), Y::float()) -> ok`

See external documentation.

```
addEllipse(This::wxGraphicsPath(), X::float(), Y::float(), W::float(),  
H::float()) -> ok
```

See external documentation.

```
addLineToPoint(This::wxGraphicsPath(), P::{X::float(), Y::float()}) -> ok
```

See external documentation.

```
addLineToPoint(This::wxGraphicsPath(), X::float(), Y::float()) -> ok
```

See external documentation.

```
addPath(This::wxGraphicsPath(), Path::wxGraphicsPath()) -> ok
```

See external documentation.

```
addQuadCurveToPoint(This::wxGraphicsPath(), Cx::float(), Cy::float(),  
X::float(), Y::float()) -> ok
```

See external documentation.

```
addRectangle(This::wxGraphicsPath(), X::float(), Y::float(), W::float(),  
H::float()) -> ok
```

See external documentation.

```
addRoundedRectangle(This::wxGraphicsPath(), X::float(), Y::float(),  
W::float(), H::float(), Radius::float()) -> ok
```

See external documentation.

```
closeSubpath(This::wxGraphicsPath()) -> ok
```

See external documentation.

```
contains(This::wxGraphicsPath(), C::{X::float(), Y::float()}) -> bool()
```

Equivalent to *contains(This, C, [])*.

```
contains(This::wxGraphicsPath(), X::float() | term(), X::float() | term()) ->  
bool()
```

See external documentation.

Alternatives:

```
contains(This::wxGraphicsPath(), X::float(), Y::float()) -> contains(This, X, Y,  
[])
```

```
contains(This::wxGraphicsPath(), C::{X::float(), Y::float()}, [Option]) ->  
bool()
```

Option = {fillStyle, integer()}

```
contains(This::wxGraphicsPath(), X::float(), Y::float(), Options::[Option]) -  
> bool()
```

Types:



**Option = {fillStyle, integer()}**

See external documentation.

```
getBox(This::wxGraphicsPath()) -> {X::float(), Y::float(), W::float(),  
H::float()}
```

See external documentation.

```
getCurrentPoint(This::wxGraphicsPath()) -> {X::float(), Y::float()}
```

See external documentation.

```
transform(This::wxGraphicsPath(), Matrix::wxGraphicsMatrix() (see module  
wxGraphicsMatrix)) -> ok
```

See external documentation.

# wxGraphicsPen

---

Erlang module

See external documentation: **wxGraphicsPen**.

This class is derived (and can use functions) from:  
*wxGraphicsObject*

## DATA TYPES

`wxGraphicsPen( )`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

# wxGraphicsRenderer

---

Erlang module

See external documentation: **wxGraphicsRenderer**.

## DATA TYPES

`wxGraphicsRenderer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getDefaultRenderer() -> wxGraphicsRenderer()`

See external documentation.

`createContext(This::wxGraphicsRenderer(), Dc::wxWindowDC() (see module wxWindowDC) | wxWindow() (see module wxWindow)) -> wxGraphicsContext() (see module wxGraphicsContext)`

See external documentation.

`createPen(This::wxGraphicsRenderer(), Pen::wxPen() (see module wxPen)) -> wxGraphicsPen() (see module wxGraphicsPen)`

See external documentation.

`createBrush(This::wxGraphicsRenderer(), Brush::wxBrush() (see module wxBrush)) -> wxGraphicsBrush() (see module wxGraphicsBrush)`

See external documentation.

`createLinearGradientBrush(This::wxGraphicsRenderer(), X1::float(), Y1::float(), X2::float(), Y2::float(), C1::colour() (see module wx), C2::colour() (see module wx)) -> wxGraphicsBrush() (see module wxGraphicsBrush)`

See external documentation.

`createRadialGradientBrush(This::wxGraphicsRenderer(), Xo::float(), Yo::float(), Xc::float(), Yc::float(), Radius::float(), OColor::colour() (see module wx), CColor::colour() (see module wx)) -> wxGraphicsBrush() (see module wxGraphicsBrush)`

See external documentation.

`createFont(This::wxGraphicsRenderer(), Font::wxFont() (see module wxFont)) -> wxGraphicsFont() (see module wxGraphicsFont)`

Equivalent to `createFont(This, Font, [])`.

```
createFont(This::wxGraphicsRenderer(), Font::wxFont() (see module wxFont),  
Options::[Option]) -> wxGraphicsFont() (see module wxGraphicsFont)
```

Types:

**Option** = {col, colour() (see module wx)}

See [external documentation](#).

```
createMatrix(This::wxGraphicsRenderer()) -> wxGraphicsMatrix() (see module  
wxGraphicsMatrix)
```

Equivalent to *createMatrix(This, [])*.

```
createMatrix(This::wxGraphicsRenderer(), Options::[Option]) ->  
wxGraphicsMatrix() (see module wxGraphicsMatrix)
```

Types:

**Option** = {a, float()} | {b, float()} | {c, float()} | {d, float()} | {tx, float()} | {ty, float()}

See [external documentation](#).

```
createPath(This::wxGraphicsRenderer()) -> wxGraphicsPath() (see module  
wxGraphicsPath)
```

See [external documentation](#).

## wxGrid

---

Erlang module

See external documentation: **wxGrid**.

This class is derived (and can use functions) from:

*wxScrolledWindow*

*wxPanel*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxGrid()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxGrid()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxGrid()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), X::integer(), Y::integer() | term()) -> wxGrid()`

See external documentation.

Alternatives:

`new(Parent::wxWindow:wxWindow(), X::integer(), Y::integer()) -> wxGrid()`  
`new(Parent, X, Y, [])`

`new(Parent::wxWindow:wxWindow(), Id::integer(), [Option]) -> wxGrid()`  
 Option = {pos, {X::integer(),Y::integer()}} | {size, {W::integer(),H::integer()}} | {style, integer()}

`new(Parent::wxWindow() (see module wxWindow), X::integer(), Y::integer(), Options::[Option]) -> wxGrid()`

Types:

**Option** = {w, integer()} | {h, integer()} | {style, integer()}

See external documentation.

`appendCols(This::wxGrid()) -> bool()`

Equivalent to `appendCols(This, [])`.

`appendCols(This::wxGrid(), Options::[Option]) -> bool()`

Types:

**Option** = {numCols, integer()} | {updateLabels, bool()}

See [external documentation](#).

**appendRows**(This::wxGrid()) -> bool()

Equivalent to *appendRows(This, [])*.

**appendRows**(This::wxGrid(), Options::[Option]) -> bool()

Types:

**Option** = {numRows, integer()} | {updateLabels, bool()}

See [external documentation](#).

**autoSize**(This::wxGrid()) -> ok

See [external documentation](#).

**autoSizeColumn**(This::wxGrid(), Col::integer()) -> ok

Equivalent to *autoSizeColumn(This, Col, [])*.

**autoSizeColumn**(This::wxGrid(), Col::integer(), Options::[Option]) -> ok

Types:

**Option** = {setAsMin, bool()}

See [external documentation](#).

**autoSizeColumns**(This::wxGrid()) -> ok

Equivalent to *autoSizeColumns(This, [])*.

**autoSizeColumns**(This::wxGrid(), Options::[Option]) -> ok

Types:

**Option** = {setAsMin, bool()}

See [external documentation](#).

**autoSizeRow**(This::wxGrid(), Row::integer()) -> ok

Equivalent to *autoSizeRow(This, Row, [])*.

**autoSizeRow**(This::wxGrid(), Row::integer(), Options::[Option]) -> ok

Types:

**Option** = {setAsMin, bool()}

See [external documentation](#).

**autoSizeRows**(This::wxGrid()) -> ok

Equivalent to *autoSizeRows(This, [])*.

**autoSizeRows**(This::wxGrid(), Options::[Option]) -> ok

Types:

**Option** = {setAsMin, bool()}

See external documentation.

**beginBatch**(This::wxGrid()) -> ok

See external documentation.

**blockToDeviceRect**(This::wxGrid(), TopLeft::{R::integer(), C::integer()}, BottomRight::{R::integer(), C::integer()}) -> {X::integer(), Y::integer(), W::integer(), H::integer()}

See external documentation.

**canDragColSize**(This::wxGrid()) -> bool()

See external documentation.

**canDragRowSize**(This::wxGrid()) -> bool()

See external documentation.

**canDragGridSize**(This::wxGrid()) -> bool()

See external documentation.

**canEnableCellControl**(This::wxGrid()) -> bool()

See external documentation.

**cellToRect**(This::wxGrid(), Coords::{R::integer(), C::integer()}) -> {X::integer(), Y::integer(), W::integer(), H::integer()}

See external documentation.

**cellToRect**(This::wxGrid(), Row::integer(), Col::integer()) -> {X::integer(), Y::integer(), W::integer(), H::integer()}

See external documentation.

**clearGrid**(This::wxGrid()) -> ok

See external documentation.

**clearSelection**(This::wxGrid()) -> ok

See external documentation.

**createGrid**(This::wxGrid(), NumRows::integer(), NumCols::integer()) -> bool()

Equivalent to *createGrid(This, NumRows, NumCols, [])*.

**createGrid**(This::wxGrid(), NumRows::integer(), NumCols::integer(), Options::[Option]) -> bool()

Types:

**Option = {selmode, WxGridSelectionModes}**

**WxGridSelectionModes = integer()**

See **external documentation**.

WxGridSelectionModes is one of ?wxGrid\_wxGridSelectCells | ?wxGrid\_wxGridSelectRows | ?wxGrid\_wxGridSelectColumns

**deleteCols(This::wxGrid()) -> bool()**

Equivalent to *deleteCols(This, [])*.

**deleteCols(This::wxGrid(), Options::[Option]) -> bool()**

Types:

**Option = {pos, integer()} | {numCols, integer()} | {updateLabels, bool()}**

See **external documentation**.

**deleteRows(This::wxGrid()) -> bool()**

Equivalent to *deleteRows(This, [])*.

**deleteRows(This::wxGrid(), Options::[Option]) -> bool()**

Types:

**Option = {pos, integer()} | {numRows, integer()} | {updateLabels, bool()}**

See **external documentation**.

**disableCellEditControl(This::wxGrid()) -> ok**

See **external documentation**.

**disableDragColSize(This::wxGrid()) -> ok**

See **external documentation**.

**disableDragGridSize(This::wxGrid()) -> ok**

See **external documentation**.

**disableDragRowSize(This::wxGrid()) -> ok**

See **external documentation**.

**enableCellEditControl(This::wxGrid()) -> ok**

Equivalent to *enableCellEditControl(This, [])*.

**enableCellEditControl(This::wxGrid(), Options::[Option]) -> ok**

Types:

**Option = {enable, bool()}**

See **external documentation**.



**enableDragColSize(This::wxGrid()) -> ok**

Equivalent to *enableDragColSize(This, [])*.

**enableDragColSize(This::wxGrid(), Options::[Option]) -> ok**

Types:

**Option = {enable, bool()}**

See **external documentation**.

**enableDragGridSize(This::wxGrid()) -> ok**

Equivalent to *enableDragGridSize(This, [])*.

**enableDragGridSize(This::wxGrid(), Options::[Option]) -> ok**

Types:

**Option = {enable, bool()}**

See **external documentation**.

**enableDragRowSize(This::wxGrid()) -> ok**

Equivalent to *enableDragRowSize(This, [])*.

**enableDragRowSize(This::wxGrid(), Options::[Option]) -> ok**

Types:

**Option = {enable, bool()}**

See **external documentation**.

**enableEditing(This::wxGrid(), Edit::bool()) -> ok**

See **external documentation**.

**enableGridLines(This::wxGrid()) -> ok**

Equivalent to *enableGridLines(This, [])*.

**enableGridLines(This::wxGrid(), Options::[Option]) -> ok**

Types:

**Option = {enable, bool()}**

See **external documentation**.

**endBatch(This::wxGrid()) -> ok**

See **external documentation**.

**fit(This::wxGrid()) -> ok**

See **external documentation**.

`forceRefresh(This::wxGrid()) -> ok`

See external documentation.

`getBatchCount(This::wxGrid()) -> integer()`

See external documentation.

`getCellAlignment(This::wxGrid(), Row::integer(), Col::integer()) -> {Horiz::integer(), Vert::integer()}`

See external documentation.

`getCellBackgroundColour(This::wxGrid(), Row::integer(), Col::integer()) -> colour() (see module wx)`

See external documentation.

`getCellEditor(This::wxGrid(), Row::integer(), Col::integer()) -> wxGridCellEditor() (see module wxGridCellEditor)`

See external documentation.

`getCellFont(This::wxGrid(), Row::integer(), Col::integer()) -> wxFont() (see module wxFont)`

See external documentation.

`getCellRenderer(This::wxGrid(), Row::integer(), Col::integer()) -> wxGridCellRenderer() (see module wxGridCellRenderer)`

See external documentation.

`getCellTextColour(This::wxGrid(), Row::integer(), Col::integer()) -> colour() (see module wx)`

See external documentation.

`getCellValue(This::wxGrid(), Coords::{R::integer(), C::integer()}) -> string()`

See external documentation.

`getCellValue(This::wxGrid(), Row::integer(), Col::integer()) -> string()`

See external documentation.

`getColLabelAlignment(This::wxGrid()) -> {Horiz::integer(), Vert::integer()}`

See external documentation.

`getColLabelSize(This::wxGrid()) -> integer()`

See external documentation.

`getColLabelValue(This::wxGrid(), Col::integer()) -> string()`

See external documentation.

`getColMinimalAcceptableWidth(This::wxGrid()) -> integer()`

See external documentation.

`getDefaultCellAlignment(This::wxGrid()) -> {Horiz::integer(),  
Vert::integer()}`

See external documentation.

`getDefaultCellBackgroundColour(This::wxGrid()) -> colour() (see module wx)`

See external documentation.

`getDefaultCellFont(This::wxGrid()) -> wxFont() (see module wxFont)`

See external documentation.

`getDefaultCellTextColour(This::wxGrid()) -> colour() (see module wx)`

See external documentation.

`getDefaultColLabelSize(This::wxGrid()) -> integer()`

See external documentation.

`getDefaultColSize(This::wxGrid()) -> integer()`

See external documentation.

`getDefaultEditor(This::wxGrid()) -> wxGridCellEditor() (see module  
wxGridCellEditor)`

See external documentation.

`getDefaultEditorForCell(This::wxGrid(), C::{R::integer(), C::integer()}) ->  
wxGridCellEditor() (see module wxGridCellEditor)`

See external documentation.

`getDefaultEditorForCell(This::wxGrid(), Row::integer(), Col::integer()) ->  
wxGridCellEditor() (see module wxGridCellEditor)`

See external documentation.

`getDefaultEditorForType(This::wxGrid(), TypeName::string()) ->  
wxGridCellEditor() (see module wxGridCellEditor)`

See external documentation.

`getDefaultRenderer(This::wxGrid()) -> wxGridCellRenderer() (see module  
wxGridCellRenderer)`

See external documentation.

`getDefaultRendererForCell(This::wxGrid(), Row::integer(), Col::integer()) -> wxGridCellRenderer()` (see module `wxGridCellRenderer`)

See external documentation.

`getDefaultRendererForType(This::wxGrid(), TypeName::string()) -> wxGridCellRenderer()` (see module `wxGridCellRenderer`)

See external documentation.

`getDefaultRowLabelSize(This::wxGrid()) -> integer()`

See external documentation.

`getDefaultRowSize(This::wxGrid()) -> integer()`

See external documentation.

`getGridCursorCol(This::wxGrid()) -> integer()`

See external documentation.

`getGridCursorRow(This::wxGrid()) -> integer()`

See external documentation.

`getGridLineColour(This::wxGrid()) -> colour()` (see module `wx`)

See external documentation.

`gridLinesEnabled(This::wxGrid()) -> bool()`

See external documentation.

`getLabelBackgroundColour(This::wxGrid()) -> colour()` (see module `wx`)

See external documentation.

`getLabelFont(This::wxGrid()) -> wxFont()` (see module `wxFont`)

See external documentation.

`getLabelTextColour(This::wxGrid()) -> colour()` (see module `wx`)

See external documentation.

`getNumberCols(This::wxGrid()) -> integer()`

See external documentation.

`getNumberRows(This::wxGrid()) -> integer()`

See external documentation.

`getOrCreateCellAttr(This::wxGrid(), Row::integer(), Col::integer()) -> wxGridCellAttr()` (see module `wxGridCellAttr`)

See external documentation.

`getRowMinimalAcceptableHeight(This::wxGrid()) -> integer()`

See external documentation.

`getRowLabelAlignment(This::wxGrid()) -> {Horiz::integer(), Vert::integer()}`

See external documentation.

`getRowLabelSize(This::wxGrid()) -> integer()`

See external documentation.

`getRowLabelValue(This::wxGrid(), Row::integer()) -> string()`

See external documentation.

`getRowSize(This::wxGrid(), Row::integer()) -> integer()`

See external documentation.

`getScrollLineX(This::wxGrid()) -> integer()`

See external documentation.

`getScrollLineY(This::wxGrid()) -> integer()`

See external documentation.

`getSelectedCells(This::wxGrid()) -> [{R::integer(), C::integer()}]`

See external documentation.

`getSelectedCols(This::wxGrid()) -> [integer()]`

See external documentation.

`getSelectedRows(This::wxGrid()) -> [integer()]`

See external documentation.

`getSelectionBackground(This::wxGrid()) -> colour()` (see module `wx`)

See external documentation.

`getSelectionBlockTopLeft(This::wxGrid()) -> [{R::integer(), C::integer()}]`

See external documentation.

`getSelectionBlockBottomRight(This::wxGrid()) -> [{R::integer(), C::integer()}]`

See external documentation.

`getSelectionForeground(This::wxGrid()) -> colour()` (see module `wx`)

See external documentation.

`getViewWidth(This::wxGrid()) -> integer()`

See external documentation.

`getGridWindow(This::wxGrid()) -> wxWindow()` (see module `wxWindow`)

See external documentation.

`getGridRowLabelWindow(This::wxGrid()) -> wxWindow()` (see module `wxWindow`)

See external documentation.

`getGridColLabelWindow(This::wxGrid()) -> wxWindow()` (see module `wxWindow`)

See external documentation.

`getGridCornerLabelWindow(This::wxGrid()) -> wxWindow()` (see module `wxWindow`)

See external documentation.

`hideCellEditControl(This::wxGrid()) -> ok`

See external documentation.

`insertCols(This::wxGrid()) -> bool()`

Equivalent to `insertCols(This, [])`.

`insertCols(This::wxGrid(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, integer()} | {numCols, integer()} | {updateLabels, bool()}

See external documentation.

`insertRows(This::wxGrid()) -> bool()`

Equivalent to `insertRows(This, [])`.

`insertRows(This::wxGrid(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, integer()} | {numRows, integer()} | {updateLabels, bool()}

See external documentation.

`isCellEditControlEnabled(This::wxGrid()) -> bool()`

See external documentation.

`isCurrentCellReadOnly(This::wxGrid()) -> bool()`

See external documentation.

**isEditable(This::wxGrid()) -> bool()**

See external documentation.

**isInSelection(This::wxGrid(), Coords::{R::integer(), C::integer()}) -> bool()**

See external documentation.

**isInSelection(This::wxGrid(), Row::integer(), Col::integer()) -> bool()**

See external documentation.

**isReadOnly(This::wxGrid(), Row::integer(), Col::integer()) -> bool()**

See external documentation.

**isSelection(This::wxGrid()) -> bool()**

See external documentation.

**isVisible(This::wxGrid(), Coords::{R::integer(), C::integer()}) -> bool()**

Equivalent to *isVisible(This, Coords, [])*.

**isVisible(This::wxGrid(), X::integer() | term(), X::integer() | term()) -> bool()**

See external documentation.

Alternatives:

**isVisible(This::wxGrid(), Row::integer(), Col::integer()) -> bool()**  
**isVisible(This, Row, Col, [])**

**isVisible(This::wxGrid(), Coords::{R::integer(), C::integer()}, [Option]) -> bool()**  
Option = {wholeCellVisible, bool()}

**isVisible(This::wxGrid(), Row::integer(), Col::integer(), Options::[Option]) -> bool()**

Types:

**Option = {wholeCellVisible, bool()}**

See external documentation.

**makeCellVisible(This::wxGrid(), Coords::{R::integer(), C::integer()}) -> ok**

See external documentation.

**makeCellVisible(This::wxGrid(), Row::integer(), Col::integer()) -> ok**

See external documentation.

**moveCursorDown(This::wxGrid(), ExpandSelection::bool()) -> bool()**

See external documentation.

`moveCursorLeft(This::wxGrid(), ExpandSelection::bool()) -> bool()`

See external documentation.

`moveCursorRight(This::wxGrid(), ExpandSelection::bool()) -> bool()`

See external documentation.

`moveCursorUp(This::wxGrid(), ExpandSelection::bool()) -> bool()`

See external documentation.

`moveCursorDownBlock(This::wxGrid(), ExpandSelection::bool()) -> bool()`

See external documentation.

`moveCursorLeftBlock(This::wxGrid(), ExpandSelection::bool()) -> bool()`

See external documentation.

`moveCursorRightBlock(This::wxGrid(), ExpandSelection::bool()) -> bool()`

See external documentation.

`moveCursorUpBlock(This::wxGrid(), ExpandSelection::bool()) -> bool()`

See external documentation.

`movePageDown(This::wxGrid()) -> bool()`

See external documentation.

`movePageUp(This::wxGrid()) -> bool()`

See external documentation.

`registerDataType(This::wxGrid(), TypeName::string(),  
Renderer::wxGridCellRenderer() (see module wxGridCellRenderer),  
Editor::wxGridCellEditor() (see module wxGridCellEditor)) -> ok`

See external documentation.

`saveEditControlValue(This::wxGrid()) -> ok`

See external documentation.

`selectAll(This::wxGrid()) -> ok`

See external documentation.

`selectBlock(This::wxGrid(), TopLeft::{R::integer(), C::integer()},  
BottomRight::{R::integer(), C::integer()}) -> ok`

Equivalent to `selectBlock(This, TopLeft, BottomRight, [])`.



```
selectBlock(This::wxGrid(), TopLeft::{R::integer(), C::integer()},
BottomRight::{R::integer(), C::integer()}, Options::[Option]) -> ok
```

Types:

**Option** = {addToSelected, bool()}

See external documentation.

```
selectBlock(This::wxGrid(), TopRow::integer(), LeftCol::integer(),
BottomRow::integer(), RightCol::integer()) -> ok
```

Equivalent to *selectBlock(This, TopRow, LeftCol, BottomRow, RightCol, [])*.

```
selectBlock(This::wxGrid(), TopRow::integer(), LeftCol::integer(),
BottomRow::integer(), RightCol::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {addToSelected, bool()}

See external documentation.

```
selectCol(This::wxGrid(), Col::integer()) -> ok
```

Equivalent to *selectCol(This, Col, [])*.

```
selectCol(This::wxGrid(), Col::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {addToSelected, bool()}

See external documentation.

```
selectRow(This::wxGrid(), Row::integer()) -> ok
```

Equivalent to *selectRow(This, Row, [])*.

```
selectRow(This::wxGrid(), Row::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {addToSelected, bool()}

See external documentation.

```
setCellAlignment(This::wxGrid(), Align::integer()) -> ok
```

See external documentation.

```
setCellAlignment(This::wxGrid(), Align::integer(), Row::integer(),
Col::integer()) -> ok
```

See external documentation.

```
setCellAlignment(This::wxGrid(), Row::integer(), Col::integer(),
Horiz::integer(), Vert::integer()) -> ok
```

See external documentation.

**setCellBackgroundColour(This::wxGrid(), Col::colour() (see module wx)) -> ok**

See [external documentation](#).

**setCellBackgroundColour(This::wxGrid(), X::integer() | term(), X::integer(),  
X::term() | integer()) -> ok**

See [external documentation](#).

Alternatives:

**setCellBackgroundColour(This::wxGrid(), Row::integer(), Col::integer(),  
Val::wx:colour()) -> ok**

**setCellBackgroundColour(This::wxGrid(), Colour::wx:colour(), Row::integer(),  
Col::integer()) -> ok**

**setCellEditor(This::wxGrid(), Row::integer(), Col::integer(),  
Editor::wxGridCellEditor() (see module wxGridCellEditor)) -> ok**

See [external documentation](#).

**setCellFont(This::wxGrid(), Row::integer(), Col::integer(), Val::wxFont()  
(see module wxFont)) -> ok**

See [external documentation](#).

**setCellRenderer(This::wxGrid(), Row::integer(), Col::integer(),  
Renderer::wxGridCellRenderer() (see module wxGridCellRenderer)) -> ok**

See [external documentation](#).

**setCellTextColour(This::wxGrid(), Col::colour() (see module wx)) -> ok**

See [external documentation](#).

**setCellTextColour(This::wxGrid(), X::integer() | term(), X::integer(),  
X::term() | integer()) -> ok**

See [external documentation](#).

Alternatives:

**setCellTextColour(This::wxGrid(), Row::integer(), Col::integer(),  
Val::wx:colour()) -> ok**

**setCellTextColour(This::wxGrid(), Val::wx:colour(), Row::integer(),  
Col::integer()) -> ok**

**setCellValue(This::wxGrid(), Coords::{R::integer(), C::integer()},  
S::string()) -> ok**

See [external documentation](#).

**setCellValue(This::wxGrid(), X::integer() | string(), X::integer(),  
X::string() | integer()) -> ok**

See [external documentation](#).

Alternatives:

```
setCellValue(This::wxGrid(), Row::integer(), Col::integer(), S::string()) ->
ok
```

```
setCellValue(This::wxGrid(), Val::string(), Row::integer(), Col::integer()) -
> ok
```

```
setColAttr(This::wxGrid(), Col::integer(), Attr::wxGridCellAttr() (see module
wxGridCellAttr)) -> ok
```

See external documentation.

```
setColFormatBool(This::wxGrid(), Col::integer()) -> ok
```

See external documentation.

```
setColFormatNumber(This::wxGrid(), Col::integer()) -> ok
```

See external documentation.

```
setColFormatFloat(This::wxGrid(), Col::integer()) -> ok
```

Equivalent to *setColFormatFloat(This, Col, [])*.

```
setColFormatFloat(This::wxGrid(), Col::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {width, integer()} | {precision, integer()}

See external documentation.

```
setColFormatCustom(This::wxGrid(), Col::integer(), TypeName::string()) -> ok
```

See external documentation.

```
setColLabelAlignment(This::wxGrid(), Horiz::integer(), Vert::integer()) -> ok
```

See external documentation.

```
setColLabelSize(This::wxGrid(), Height::integer()) -> ok
```

See external documentation.

```
setColLabelValue(This::wxGrid(), Col::integer(), Val::string()) -> ok
```

See external documentation.

```
setColMinimalWidth(This::wxGrid(), Col::integer(), Width::integer()) -> ok
```

See external documentation.

```
setColMinimalAcceptableWidth(This::wxGrid(), Width::integer()) -> ok
```

See external documentation.

```
setColSize(This::wxGrid(), Col::integer(), Width::integer()) -> ok
```

See external documentation.

`setDefaultCellAlignment(This::wxGrid(), Horiz::integer(), Vert::integer()) -> ok`

See [external documentation](#).

`setDefaultCellBackgroundColour(This::wxGrid(), Val::colour() (see module wx)) -> ok`

See [external documentation](#).

`setDefaultCellFont(This::wxGrid(), Val::wxFont() (see module wxFont)) -> ok`

See [external documentation](#).

`setDefaultCellTextColour(This::wxGrid(), Val::colour() (see module wx)) -> ok`

See [external documentation](#).

`setDefaultEditor(This::wxGrid(), Editor::wxGridCellEditor() (see module wxGridCellEditor)) -> ok`

See [external documentation](#).

`setDefaultRenderer(This::wxGrid(), Renderer::wxGridCellRenderer() (see module wxGridCellRenderer)) -> ok`

See [external documentation](#).

`setDefaultColSize(This::wxGrid(), Width::integer()) -> ok`

Equivalent to `setDefaultColSize(This, Width, [])`.

`setDefaultColSize(This::wxGrid(), Width::integer(), Options::[Option]) -> ok`

Types:

**Option** = {resizeExistingCols, bool()}

See [external documentation](#).

`setDefaultRowSize(This::wxGrid(), Height::integer()) -> ok`

Equivalent to `setDefaultRowSize(This, Height, [])`.

`setDefaultRowSize(This::wxGrid(), Height::integer(), Options::[Option]) -> ok`

Types:

**Option** = {resizeExistingRows, bool()}

See [external documentation](#).

`setGridCursor(This::wxGrid(), Row::integer(), Col::integer()) -> ok`

See [external documentation](#).

`setGridLineColour(This::wxGrid(), Val::colour() (see module wx)) -> ok`

See [external documentation](#).

`setLabelBackgroundColour(This::wxGrid(), Val::colour() (see module wx)) -> ok`  
See external documentation.

`setLabelFont(This::wxGrid(), Val::wxFont() (see module wxFont)) -> ok`  
See external documentation.

`setLabelTextColour(This::wxGrid(), Val::colour() (see module wx)) -> ok`  
See external documentation.

`setMargins(This::wxGrid(), ExtraWidth::integer(), ExtraHeight::integer()) -> ok`  
See external documentation.

`setReadOnly(This::wxGrid(), Row::integer(), Col::integer()) -> ok`  
Equivalent to `setReadOnly(This, Row, Col, [])`.

`setReadOnly(This::wxGrid(), Row::integer(), Col::integer(), Options::[Option]) -> ok`  
Types:

Option = {isReadOnly, bool()}

See external documentation.

`setRowAttr(This::wxGrid(), Row::integer(), Attr::wxGridCellAttr() (see module wxGridCellAttr)) -> ok`  
See external documentation.

`setRowLabelAlignment(This::wxGrid(), Horiz::integer(), Vert::integer()) -> ok`  
See external documentation.

`setRowLabelSize(This::wxGrid(), Width::integer()) -> ok`  
See external documentation.

`setRowLabelValue(This::wxGrid(), Row::integer(), Val::string()) -> ok`  
See external documentation.

`setRowMinimalHeight(This::wxGrid(), Row::integer(), Width::integer()) -> ok`  
See external documentation.

`setRowMinimalAcceptableHeight(This::wxGrid(), Width::integer()) -> ok`  
See external documentation.

`setRowSize(This::wxGrid(), Row::integer(), Height::integer()) -> ok`  
See external documentation.

**setScrollLineX(This::wxGrid(), X::integer()) -> ok**

See [external documentation](#).

**setScrollLineY(This::wxGrid(), Y::integer()) -> ok**

See [external documentation](#).

**setSelectionBackground(This::wxGrid(), C::colour() (see module wx)) -> ok**

See [external documentation](#).

**setSelectionForeground(This::wxGrid(), C::colour() (see module wx)) -> ok**

See [external documentation](#).

**setSelectionMode(This::wxGrid(), Selmode::WxGridSelectionMode) -> ok**

Types:

**WxGridSelectionMode = integer()**

See [external documentation](#).

WxGridSelectionMode is one of ?wxGrid\_wxGridSelectCells | ?wxGrid\_wxGridSelectRows | ?wxGrid\_wxGridSelectColumns

**showCellEditControl(This::wxGrid()) -> ok**

See [external documentation](#).

**xToCol(This::wxGrid(), X::integer()) -> integer()**

Equivalent to *xToCol(This, X, [])*.

**xToCol(This::wxGrid(), X::integer(), Options::[Option]) -> integer()**

Types:

**Option = {clipToMinMax, bool()}**

See [external documentation](#).

**xToEdgeOfCol(This::wxGrid(), X::integer()) -> integer()**

See [external documentation](#).

**yToEdgeOfRow(This::wxGrid(), Y::integer()) -> integer()**

See [external documentation](#).

**yToRow(This::wxGrid(), Y::integer()) -> integer()**

See [external documentation](#).

**destroy(This::wxGrid()) -> ok**

Destroys this object, do not use object again

# wxGridBagSizer

Erlang module

See external documentation: **wxGridBagSizer**.

This class is derived (and can use functions) from:

*wxFlexGridSizer*

*wxGridSizer*

*wxSizer*

## DATA TYPES

`wxGridBagSizer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**`new()` -> `wxGridBagSizer()`**

Equivalent to `new([])`.

**`new(Options::[Option])` -> `wxGridBagSizer()`**

Types:

**Option** = {vgap, integer()} | {hgap, integer()}

See external documentation.

**`add(This::wxGridBagSizer(), Item::wxSizerItem() (see module wxSizerItem) | wxGBSizerItem() (see module wxGBSizerItem))` -> `wxSizerItem() (see module wxSizerItem)`**

See external documentation.

**`add(This::wxGridBagSizer(), X::integer() | term(), X::integer() | term())` -> `wxSizerItem() (see module wxSizerItem)`**

See external documentation.

Alternatives:

`add(This::wxGridBagSizer(), Width::integer(), Height::integer())` -> `add(This,Width,Height, [])`

`add(This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer(), Pos::{R::integer(),C::integer()})` -> `add(This,Window,Pos, [])`

`add(This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer(), [Option])` -> `wxSizerItem:wxSizerItem()`

**Option** = {proportion, integer()} | {flag, integer()} | {border, integer()} | {userData, wx:wx()}

```
add(This::wxGridBagSizer(), X::integer() | term(), X::integer() | term(),
X::term()) -> wxSizerItem() (see module wxSizerItem)
```

See [external documentation](#).

Alternatives:

```
add(This::wxGridBagSizer(), Width::integer(), Height::integer(), Pos::
{R::integer(),C::integer()}) -> add(This,Width,Height,Pos, [])
```

```
add(This::wxGridBagSizer(), Width::integer(), Height::integer(), [Option]) -
> wxSizerItem:wxSizerItem()
```

Option = {proportion, integer()} | {flag, integer()} | {border, integer()} | {userData, wx:wx()}

```
add(This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer(),
Pos::{R::integer(),C::integer()}, [Option]) -> wxSizerItem:wxSizerItem()
```

Option = {span, {RS::integer(),CS::integer()}} | {flag, integer()} | {border, integer()} | {userData, wx:wx()}

```
add(This::wxGridBagSizer(), Width::integer(), Height::integer(), Pos::
{R::integer(), C::integer()}, Options::[Option]) -> wxSizerItem() (see module
wxSizerItem)
```

Types:

Option = {span, {RS::integer(), CS::integer()}} | {flag, integer()} | {border, integer()} | {userData, wx()}  
(see module wx)

See [external documentation](#).

```
calcMin(This::wxGridBagSizer()) -> {W::integer(), H::integer()}
```

See [external documentation](#).

```
checkForIntersection(This::wxGridBagSizer(), Item::wxGBSizerItem() (see
module wxGBSizerItem)) -> bool()
```

Equivalent to *checkForIntersection(This, Item, [])*.

```
checkForIntersection(This::wxGridBagSizer(), X::term(), X::term()) -> bool()
```

See [external documentation](#).

Alternatives:

```
checkForIntersection(This::wxGridBagSizer(), Pos::
{R::integer(),C::integer()}, Span::{RS::integer(),CS::integer()}) ->
checkForIntersection(This,Pos,Span, [])
```

```
checkForIntersection(This::wxGridBagSizer(),
Item::wxGBSizerItem:wxGBSizerItem(), [Option]) -> bool()
```

Option = {excludeItem, wxGBSizerItem:wxGBSizerItem()}

```
checkForIntersection(This::wxGridBagSizer(), Pos::{R::integer(),
C::integer()}, Span::{RS::integer(), CS::integer()}, Options::[Option]) ->
bool()
```

Types:

Option = {excludeItem, wxGBSizerItem() (see module wxGBSizerItem)}

See [external documentation](#).



```
findItem(This::wxGridBagSizer(), Window::wxWindow() (see module wxWindow) |  
wxSizer() (see module wxSizer)) -> wxGBSizerItem() (see module wxGBSizerItem)
```

See external documentation.

```
findItemAtPoint(This::wxGridBagSizer(), Pt::{X::integer(), Y::integer()}) ->  
wxGBSizerItem() (see module wxGBSizerItem)
```

See external documentation.

```
findItemAtPosition(This::wxGridBagSizer(), Pos::{R::integer(), C::integer()})  
-> wxGBSizerItem() (see module wxGBSizerItem)
```

See external documentation.

```
findItemWithData(This::wxGridBagSizer(), UserData::wx() (see module wx)) ->  
wxGBSizerItem() (see module wxGBSizerItem)
```

See external documentation.

```
getCellSize(This::wxGridBagSizer(), Row::integer(), Col::integer()) ->  
{W::integer(), H::integer()}
```

See external documentation.

```
getEmptyCellSize(This::wxGridBagSizer()) -> {W::integer(), H::integer()}
```

See external documentation.

```
getItemPosition(This::wxGridBagSizer(), X::integer() | term()) ->  
{R::integer(), C::integer()}
```

See external documentation.

Alternatives:

```
getItemPosition(This::wxGridBagSizer(), Index::integer()) ->  
{R::integer(), C::integer()}
```

```
getItemPosition(This::wxGridBagSizer(), Window::wxWindow:wxWindow() |  
wxSizer:wxSizer()) -> {R::integer(), C::integer()}
```

```
getItemSpan(This::wxGridBagSizer(), X::integer() | term()) -> {RS::integer(),  
CS::integer()}
```

See external documentation.

Alternatives:

```
getItemSpan(This::wxGridBagSizer(), Index::integer()) ->  
{RS::integer(), CS::integer()}
```

```
getItemSpan(This::wxGridBagSizer(), Window::wxWindow:wxWindow() |  
wxSizer:wxSizer()) -> {RS::integer(), CS::integer()}
```

```
setEmptyCellSize(This::wxGridBagSizer(), Sz::{W::integer(), H::integer()}) ->  
ok
```

See external documentation.

```
setItemPosition(This::wxGridBagSizer(), X::integer() | term(), Pos::
{R::integer(), C::integer()}) -> bool()
```

See [external documentation](#).

Alternatives:

```
setItemPosition(This::wxGridBagSizer(),      Index::integer(),      Pos::
{R::integer(),C::integer()}) -> bool()
```

```
setItemPosition(This::wxGridBagSizer(),      Window::wxWindow:wxWindow()      |
wxSizer:wxSizer(), Pos::{R::integer(),C::integer()}) -> bool()
```

```
setItemSpan(This::wxGridBagSizer(), X::integer() | term(), Span::
{RS::integer(), CS::integer()}) -> bool()
```

See [external documentation](#).

Alternatives:

```
setItemSpan(This::wxGridBagSizer(),      Index::integer(),      Span::
{RS::integer(),CS::integer()}) -> bool()
```

```
setItemSpan(This::wxGridBagSizer(),      Window::wxWindow:wxWindow()      |
wxSizer:wxSizer(), Span::{RS::integer(),CS::integer()}) -> bool()
```

```
destroy(This::wxGridBagSizer()) -> ok
```

Destroys this object, do not use object again

## wxGridCellAttr

---

Erlang module

See external documentation: **wxGridCellAttr**.

### DATA TYPES

`wxGridCellAttr()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`setTextColour(This::wxGridCellAttr(), ColText::colour() (see module wx)) -> ok`

See external documentation.

`setBackgroundColour(This::wxGridCellAttr(), ColBack::colour() (see module wx)) -> ok`

See external documentation.

`setFont(This::wxGridCellAttr(), Font::wxFont() (see module wxFont)) -> ok`

See external documentation.

`setAlignment(This::wxGridCellAttr(), HAlign::integer(), VAlign::integer()) -> ok`

See external documentation.

`setReadOnly(This::wxGridCellAttr()) -> ok`

Equivalent to `setReadOnly(This, [])`.

`setReadOnly(This::wxGridCellAttr(), Options::[Option]) -> ok`

Types:

**Option** = {isReadOnly, bool()}

See external documentation.

`setRenderer(This::wxGridCellAttr(), Renderer::wxGridCellRenderer() (see module wxGridCellRenderer)) -> ok`

See external documentation.

`setEditor(This::wxGridCellAttr(), Editor::wxGridCellEditor() (see module wxGridCellEditor)) -> ok`

See external documentation.

`hasTextColour(This::wxGridCellAttr()) -> bool()`

See external documentation.

`hasBackgroundColour(This::wxGridCellAttr()) -> bool()`

See external documentation.

`hasFont(This::wxGridCellAttr()) -> bool()`

See external documentation.

`hasAlignment(This::wxGridCellAttr()) -> bool()`

See external documentation.

`hasRenderer(This::wxGridCellAttr()) -> bool()`

See external documentation.

`hasEditor(This::wxGridCellAttr()) -> bool()`

See external documentation.

`getTextColour(This::wxGridCellAttr()) -> colour() (see module wx)`

See external documentation.

`getBackgroundColour(This::wxGridCellAttr()) -> colour() (see module wx)`

See external documentation.

`getFont(This::wxGridCellAttr()) -> wxFont() (see module wxFont)`

See external documentation.

`getAlignment(This::wxGridCellAttr()) -> {HAlign::integer(),  
VAlign::integer()}`

See external documentation.

`getRenderer(This::wxGridCellAttr(), Grid::wxGrid() (see module wxGrid),  
Row::integer(), Col::integer()) -> wxGridCellRenderer() (see module  
wxGridCellRenderer)`

See external documentation.

`getEditor(This::wxGridCellAttr(), Grid::wxGrid() (see module wxGrid),  
Row::integer(), Col::integer()) -> wxGridCellEditor() (see module  
wxGridCellEditor)`

See external documentation.

`isReadOnly(This::wxGridCellAttr()) -> bool()`

See external documentation.

```
setDefAttr(This::wxGridCellAttr(), DefAttr::wxGridCellAttr()) -> ok
```

See external documentation.

## wxGridCellBoolEditor

---

Erlang module

See external documentation: **wxGridCellBoolEditor**.

This class is derived (and can use functions) from:  
*wxGridCellEditor*

### DATA TYPES

`wxGridCellBoolEditor()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxGridCellBoolEditor()`**

See **external documentation**.

**`isTrueValue(Value::string())`** -> **`bool()`**

See **external documentation**.

**`useStringValues()`** -> **`ok`**

Equivalent to *useStringValues([])*.

**`useStringValues(Options::[Option])`** -> **`ok`**

Types:

**`Option = {valueTrue, string()} | {valueFalse, string()}`**

See **external documentation**.

**`destroy(This::wxGridCellBoolEditor())`** -> **`ok`**

Destroys this object, do not use object again

## wxGridCellBoolRenderer

---

Erlang module

See external documentation: **wxGridCellBoolRenderer**.

This class is derived (and can use functions) from:  
*wxGridCellRenderer*

### DATA TYPES

`wxGridCellBoolRenderer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxGridCellBoolRenderer()`**

See external documentation.

**`destroy(This::wxGridCellBoolRenderer())`** -> **`ok`**

Destroys this object, do not use object again

## wxGridCellChoiceEditor

---

Erlang module

See external documentation: **wxGridCellChoiceEditor**.

This class is derived (and can use functions) from:  
*wxGridCellEditor*

### DATA TYPES

`wxGridCellChoiceEditor()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Choices::[[string()]]) -> wxGridCellChoiceEditor()`

Equivalent to `new(Choices, [])`.

`new(Choices::[[string()]], Options::[Option]) -> wxGridCellChoiceEditor()`

Types:

**Option** = {allowOthers, bool()}

See external documentation.

`setParameters(This::wxGridCellChoiceEditor(), Params::string()) -> ok`

See external documentation.

`destroy(This::wxGridCellChoiceEditor()) -> ok`

Destroys this object, do not use object again



## wxGridCellEditor

---

Erlang module

See external documentation: **wxGridCellEditor**.

### DATA TYPES

`wxGridCellEditor()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`create(This::wxGridCellEditor(), Parent::wxWindow() (see module wxWindow), Id::integer(), EvtHandler::wxEvtHandler() (see module wxEvtHandler)) -> ok`

See external documentation.

`isCreated(This::wxGridCellEditor()) -> bool()`

See external documentation.

`setSize(This::wxGridCellEditor(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> ok`

See external documentation.

`show(This::wxGridCellEditor(), Show::bool()) -> ok`

Equivalent to `show(This, Show, [])`.

`show(This::wxGridCellEditor(), Show::bool(), Options::[Option]) -> ok`

Types:

`Option = {attr, wxGridCellAttr() (see module wxGridCellAttr)}`

See external documentation.

`paintBackground(This::wxGridCellEditor(), RectCell::{X::integer(), Y::integer(), W::integer(), H::integer()}, Attr::wxGridCellAttr() (see module wxGridCellAttr)) -> ok`

See external documentation.

`beginEdit(This::wxGridCellEditor(), Row::integer(), Col::integer(), Grid::wxGrid() (see module wxGrid)) -> ok`

See external documentation.

`endEdit(This::wxGridCellEditor(), Row::integer(), Col::integer(), Grid::wxGrid() (see module wxGrid)) -> bool()`

See external documentation.

`reset(This::wxGridCellEditor()) -> ok`

See external documentation.

`startingKey(This::wxGridCellEditor(), Event::wxKeyEvent() (see module wxKeyEvent)) -> ok`

See external documentation.

`startingClick(This::wxGridCellEditor()) -> ok`

See external documentation.

`handleReturn(This::wxGridCellEditor(), Event::wxKeyEvent() (see module wxKeyEvent)) -> ok`

See external documentation.

## wxGridCellFloatEditor

---

Erlang module

See external documentation: **wxGridCellFloatEditor**.

This class is derived (and can use functions) from:  
*wxGridCellEditor*

### DATA TYPES

`wxGridCellFloatEditor()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxGridCellFloatEditor()`**

Equivalent to *new([])*.

**`new(Options::[Option])`** -> **`wxGridCellFloatEditor()`**

Types:

**`Option = {width, integer()} | {precision, integer()}`**

See external documentation.

**`setParameters(This::wxGridCellFloatEditor(), Params::string())`** -> **`ok`**

See external documentation.

**`destroy(This::wxGridCellFloatEditor())`** -> **`ok`**

Destroys this object, do not use object again

## wxGridCellFloatRenderer

---

Erlang module

See external documentation: **wxGridCellFloatRenderer**.

This class is derived (and can use functions) from:

*wxGridCellStringRenderer*

*wxGridCellRenderer*

### DATA TYPES

`wxGridCellFloatRenderer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxGridCellFloatRenderer()`

Equivalent to `new([])`.

`new(Options:::[Option]) -> wxGridCellFloatRenderer()`

Types:

**Option** = {width, integer()} | {precision, integer()}

See external documentation.

`getPrecision(This::wxGridCellFloatRenderer()) -> integer()`

See external documentation.

`getWidth(This::wxGridCellFloatRenderer()) -> integer()`

See external documentation.

`setParameters(This::wxGridCellFloatRenderer(), Params::string()) -> ok`

See external documentation.

`setPrecision(This::wxGridCellFloatRenderer(), Precision::integer()) -> ok`

See external documentation.

`setWidth(This::wxGridCellFloatRenderer(), Width::integer()) -> ok`

See external documentation.

`destroy(This::wxGridCellFloatRenderer()) -> ok`

Destroys this object, do not use object again

## wxGridCellNumberEditor

---

Erlang module

See external documentation: **wxGridCellNumberEditor**.

This class is derived (and can use functions) from:

*wxGridCellTextEditor*

*wxGridCellEditor*

### DATA TYPES

`wxGridCellNumberEditor()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxGridCellNumberEditor()`**

Equivalent to *new([])*.

**`new(Options::[Option])`** -> **`wxGridCellNumberEditor()`**

Types:

**`Option = {min, integer()} | {max, integer()}`**

See external documentation.

**`getValue(This::wxGridCellNumberEditor())`** -> **`string()`**

See external documentation.

**`setParameters(This::wxGridCellNumberEditor(), Params::string())`** -> **`ok`**

See external documentation.

**`destroy(This::wxGridCellNumberEditor())`** -> **`ok`**

Destroys this object, do not use object again

## wxGridCellNumberRenderer

---

Erlang module

See external documentation: **wxGridCellNumberRenderer**.

This class is derived (and can use functions) from:

*wxGridCellStringRenderer*

*wxGridCellRenderer*

### DATA TYPES

`wxGridCellNumberRenderer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxGridCellNumberRenderer()`**

See **external documentation**.

**`destroy(This::wxGridCellNumberRenderer())`** -> **`ok`**

Destroys this object, do not use object again

## wxGridCellRenderer

---

Erlang module

See external documentation: **wxGridCellRenderer**.

### DATA TYPES

`wxGridCellRenderer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

```
draw(This::wxGridCellRenderer(), Grid::wxGrid() (see module wxGrid),
Attr::wxGridCellAttr() (see module wxGridCellAttr), Dc::wxDC() (see module
wxDC), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()},
Row::integer(), Col::integer(), IsSelected::bool()) -> ok
```

See external documentation.

```
getBestSize(This::wxGridCellRenderer(), Grid::wxGrid() (see module wxGrid),
Attr::wxGridCellAttr() (see module wxGridCellAttr), Dc::wxDC() (see module
wxDC), Row::integer(), Col::integer()) -> {W::integer(), H::integer()}
```

See external documentation.

# wxGridCellStringRenderer

---

Erlang module

See external documentation: **wxGridCellStringRenderer**.

This class is derived (and can use functions) from:  
*wxGridCellRenderer*

## DATA TYPES

`wxGridCellStringRenderer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**`new()`** -> **`wxGridCellStringRenderer()`**

See **external documentation**.

**`destroy(This::wxGridCellStringRenderer())`** -> **`ok`**

Destroys this object, do not use object again



## wxGridCellTextEditor

---

Erlang module

See external documentation: **wxGridCellTextEditor**.

This class is derived (and can use functions) from:  
*wxGridCellEditor*

### DATA TYPES

`wxGridCellTextEditor()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxGridCellTextEditor()`**

See external documentation.

**`setParameters(This::wxGridCellTextEditor(), Params::string())`** -> **`ok`**

See external documentation.

**`destroy(This::wxGridCellTextEditor())`** -> **`ok`**

Destroys this object, do not use object again

## wxGridEvent

---

Erlang module

See external documentation: **wxGridEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*grid\_cell\_left\_click, grid\_cell\_right\_click, grid\_cell\_left\_dclick, grid\_cell\_right\_dclick, grid\_label\_left\_click, grid\_label\_right\_click, grid\_label\_left\_dclick, grid\_label\_right\_dclick, grid\_row\_size, grid\_col\_size, grid\_range\_select, grid\_cell\_change, grid\_select\_cell, grid\_editor\_shown, grid\_editor\_hidden, grid\_editor\_created, grid\_cell\_begin\_drag*

See also the message variant `#wxGrid{}` event record type.

This class is derived (and can use functions) from:

*wxNotifyEvent*

*wxCommandEvent*

*wxEvent*

## DATA TYPES

`wxGridEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`altDown(This::wxGridEvent()) -> bool()`

See external documentation.

`controlDown(This::wxGridEvent()) -> bool()`

See external documentation.

`getCol(This::wxGridEvent()) -> integer()`

See external documentation.

`getPosition(This::wxGridEvent()) -> {X::integer(), Y::integer()}`

See external documentation.

`getRow(This::wxGridEvent()) -> integer()`

See external documentation.

`metaDown(This::wxGridEvent()) -> bool()`

See external documentation.

`selecting(This::wxGridEvent()) -> bool()`

See external documentation.

`shiftDown(This::wxGridEvent()) -> bool()`

See external documentation.

## wxGridSizer

---

Erlang module

See external documentation: **wxGridSizer**.

This class is derived (and can use functions) from:  
*wxSizer*

### DATA TYPES

`wxGridSizer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Cols::integer()) -> wxGridSizer()`

Equivalent to `new(Cols, [])`.

`new(Cols::integer(), Options::[Option]) -> wxGridSizer()`

Types:

**Option** = {vgap, integer()} | {hgap, integer()}

See external documentation.

`new(Rows::integer(), Cols::integer(), Vgap::integer(), Hgap::integer()) -> wxGridSizer()`

See external documentation.

`getCols(This::wxGridSizer()) -> integer()`

See external documentation.

`getHGap(This::wxGridSizer()) -> integer()`

See external documentation.

`getRows(This::wxGridSizer()) -> integer()`

See external documentation.

`getVGap(This::wxGridSizer()) -> integer()`

See external documentation.

`setCols(This::wxGridSizer(), Cols::integer()) -> ok`

See external documentation.

**setHGap(This::wxGridSizer(), Gap::integer()) -> ok**

See external documentation.

**setRows(This::wxGridSizer(), Rows::integer()) -> ok**

See external documentation.

**setVGap(This::wxGridSizer(), Gap::integer()) -> ok**

See external documentation.

**destroy(This::wxGridSizer()) -> ok**

Destroys this object, do not use object again

## wxHelpEvent

---

Erlang module

See external documentation: **wxHelpEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*help, detailed\_help*

See also the message variant *#wxHelp{}* event record type.

This class is derived (and can use functions) from:

*wxEvt*

## DATA TYPES

`wxHelpEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

`getOrigin(This::wxHelpEvent()) -> Origin`

Types:

**Origin = integer()**

See **external documentation**.

Origin is one of ?wxHelpEvent\_Origin\_Unknown | ?wxHelpEvent\_Origin\_Keyboard | ?wxHelpEvent\_Origin\_HelpButton

`getPosition(This::wxHelpEvent()) -> {X::integer(), Y::integer()}`

See **external documentation**.

`setOrigin(This::wxHelpEvent(), Origin) -> ok`

Types:

**Origin = integer()**

See **external documentation**.

Origin is one of ?wxHelpEvent\_Origin\_Unknown | ?wxHelpEvent\_Origin\_Keyboard | ?wxHelpEvent\_Origin\_HelpButton

`setPosition(This::wxHelpEvent(), Pos::{X::integer(), Y::integer()}) -> ok`

See **external documentation**.

# wxHtmlEasyPrinting

---

Erlang module

See external documentation: [wxHtmlEasyPrinting](#).

## DATA TYPES

`wxHtmlEasyPrinting()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxHtmlEasyPrinting()`

Equivalent to `new([])`.

`new(Options:::[Option]) -> wxHtmlEasyPrinting()`

Types:

`Option = {name, string()} | {parentWindow, wxWindow()} (see module wxWindow)`

See external documentation.

`getPrintData(This::wxHtmlEasyPrinting()) -> wxPrintData() (see module wxPrintData)`

See external documentation.

`getPageSetupData(This::wxHtmlEasyPrinting()) -> wxPageSetupDialogData() (see module wxPageSetupDialogData)`

See external documentation.

`previewFile(This::wxHtmlEasyPrinting(), Htmlfile::string()) -> bool()`

See external documentation.

`previewText(This::wxHtmlEasyPrinting(), Htmltext::string()) -> bool()`

Equivalent to `previewText(This, Htmltext, [])`.

`previewText(This::wxHtmlEasyPrinting(), Htmltext::string(), Options:::[Option]) -> bool()`

Types:

`Option = {basepath, string() }`

See external documentation.

`printFile(This::wxHtmlEasyPrinting(), Htmlfile::string()) -> bool()`

See external documentation.

```
printText(This::wxHtmlEasyPrinting(), Htmltext::string()) -> bool()
```

Equivalent to *printText(This, Htmltext, [])*.

```
printText(This::wxHtmlEasyPrinting(), Htmltext::string(), Options::[Option])  
-> bool()
```

Types:

**Option** = {basepath, string()}

See [external documentation](#).

```
pageSetup(This::wxHtmlEasyPrinting()) -> ok
```

See [external documentation](#).

```
setFonts(This::wxHtmlEasyPrinting(), Normal_face::string(),  
Fixed_face::string()) -> ok
```

Equivalent to *setFonts(This, Normal\_face, Fixed\_face, [])*.

```
setFonts(This::wxHtmlEasyPrinting(), Normal_face::string(),  
Fixed_face::string(), Options::[Option]) -> ok
```

Types:

**Option** = {sizes, [integer()]}

See [external documentation](#).

```
setHeader(This::wxHtmlEasyPrinting(), Header::string()) -> ok
```

Equivalent to *setHeader(This, Header, [])*.

```
setHeader(This::wxHtmlEasyPrinting(), Header::string(), Options::[Option]) ->  
ok
```

Types:

**Option** = {pg, integer()}

See [external documentation](#).

```
setFooter(This::wxHtmlEasyPrinting(), Footer::string()) -> ok
```

Equivalent to *setFooter(This, Footer, [])*.

```
setFooter(This::wxHtmlEasyPrinting(), Footer::string(), Options::[Option]) ->  
ok
```

Types:

**Option** = {pg, integer()}

See [external documentation](#).

```
destroy(This::wxHtmlEasyPrinting()) -> ok
```

Destroys this object, do not use object again



## wxHtmlLinkEvent

---

Erlang module

See external documentation: **wxHtmlLinkEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*command\_html\_link\_clicked*

See also the message variant *#wxHtmlLink{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

### DATA TYPES

`wxHtmlLinkEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

`getLinkInfo(This::wxHtmlLinkEvent()) -> wxHtmlLinkInfo()` (see module `wx`)

See **external documentation**.

## wxHtmlWindow

---

Erlang module

See external documentation: **wxHtmlWindow**.

This class is derived (and can use functions) from:

*wxScrolledWindow*

*wxPanel*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxHtmlWindow()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxHtmlWindow()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxHtmlWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxHtmlWindow()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`appendToPage(This::wxHtmlWindow(), Source::string()) -> bool()`

See external documentation.

`getOpenedAnchor(This::wxHtmlWindow()) -> string()`

See external documentation.

`getOpenedPage(This::wxHtmlWindow()) -> string()`

See external documentation.

`getOpenedPageTitle(This::wxHtmlWindow()) -> string()`

See external documentation.

`getRelatedFrame(This::wxHtmlWindow()) -> wxFrame()` (see module `wxFrame`)

See external documentation.

`historyBack(This::wxHtmlWindow()) -> bool()`

See external documentation.

`historyCanBack(This::wxHtmlWindow()) -> bool()`

See external documentation.

`historyCanForward(This::wxHtmlWindow()) -> bool()`

See external documentation.

`historyClear(This::wxHtmlWindow()) -> ok`

See external documentation.

`historyForward(This::wxHtmlWindow()) -> bool()`

See external documentation.

`loadFile(This::wxHtmlWindow(), Filename::string()) -> bool()`

See external documentation.

`loadPage(This::wxHtmlWindow(), Location::string()) -> bool()`

See external documentation.

`selectAll(This::wxHtmlWindow()) -> ok`

See external documentation.

`selectionToText(This::wxHtmlWindow()) -> string()`

See external documentation.

`selectLine(This::wxHtmlWindow(), Pos::{X::integer(), Y::integer()}) -> ok`

See external documentation.

`selectWord(This::wxHtmlWindow(), Pos::{X::integer(), Y::integer()}) -> ok`

See external documentation.

`setBorders(This::wxHtmlWindow(), B::integer()) -> ok`

See external documentation.

`setFonts(This::wxHtmlWindow(), Normal_face::string(), Fixed_face::string()) -> ok`

Equivalent to `setFonts(This, Normal_face, Fixed_face, [])`.

```
setFont(This::wxHtmlWindow(), Normal_face::string(), Fixed_face::string(),  
Options::[Option]) -> ok
```

Types:

**Option** = {sizes, integer()}

See external documentation.

```
setPage(This::wxHtmlWindow(), Source::string()) -> bool()
```

See external documentation.

```
setRelatedFrame(This::wxHtmlWindow(), Frame::wxFrame() (see module wxFrame),  
Format::string()) -> ok
```

See external documentation.

```
setRelatedStatusBar(This::wxHtmlWindow(), Bar::integer()) -> ok
```

See external documentation.

```
toText(This::wxHtmlWindow()) -> string()
```

See external documentation.

```
destroy(This::wxHtmlWindow()) -> ok
```

Destroys this object, do not use object again

## wxIcon

Erlang module

See external documentation: **wxIcon**.

This class is derived (and can use functions) from:

*wxBitmap*

## DATA TYPES

`wxIcon()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxIcon()`

See external documentation.

`new(X::string() | term()) -> wxIcon()`

See external documentation.

Alternatives:

`new(Filename::string()) -> new(Filename, [])`

`new(Loc::wx:wx()) -> wxIcon()`

`new(Filename::string(), Options::[Option]) -> wxIcon()`

Types:

**Option** = {type, WxBitmapType} | {desiredWidth, integer()} | {desiredHeight, integer()}

**WxBitmapType** = integer()

See external documentation.

WxBitmapType is one of ?wxBITMAP\_TYPE\_INVALID | ?wxBITMAP\_TYPE\_BMP | ?wxBITMAP\_TYPE\_BMP\_RESOURCE | ?wxBITMAP\_TYPE\_RESOURCE | ?wxBITMAP\_TYPE\_ICO | ?wxBITMAP\_TYPE\_ICO\_RESOURCE | ?wxBITMAP\_TYPE\_CUR | ?wxBITMAP\_TYPE\_CUR\_RESOURCE | ?wxBITMAP\_TYPE\_XBM | ?wxBITMAP\_TYPE\_XBM\_DATA | ?wxBITMAP\_TYPE\_XPM | ?wxBITMAP\_TYPE\_XPM\_DATA | ?wxBITMAP\_TYPE\_TIF | ?wxBITMAP\_TYPE\_TIF\_RESOURCE | ?wxBITMAP\_TYPE\_GIF | ?wxBITMAP\_TYPE\_GIF\_RESOURCE | ?wxBITMAP\_TYPE\_PNG | ?wxBITMAP\_TYPE\_PNG\_RESOURCE | ?wxBITMAP\_TYPE\_JPEG | ?wxBITMAP\_TYPE\_JPEG\_RESOURCE | ?wxBITMAP\_TYPE\_PNM | ?wxBITMAP\_TYPE\_PNM\_RESOURCE | ?wxBITMAP\_TYPE\_PCX | ?wxBITMAP\_TYPE\_PCX\_RESOURCE | ?wxBITMAP\_TYPE\_PICT | ?wxBITMAP\_TYPE\_PICT\_RESOURCE | ?wxBITMAP\_TYPE\_ICON | ?wxBITMAP\_TYPE\_ICON\_RESOURCE | ?wxBITMAP\_TYPE\_ANI | ?wxBITMAP\_TYPE\_IFF | ?wxBITMAP\_TYPE\_TGA | ?wxBITMAP\_TYPE\_MACCURSOR | ?wxBITMAP\_TYPE\_MACCURSOR\_RESOURCE | ?wxBITMAP\_TYPE\_ANY

`copyFromBitmap(This::wxIcon(), Bmp::wxBitmap() (see module wxBitmap)) -> ok`

See external documentation.

## **wxIcon**

---

**`destroy(This::wxIcon()) -> ok`**

Destroys this object, do not use object again

## wxIconBundle

---

Erlang module

See external documentation: **wxIconBundle**.

### DATA TYPES

`wxIconBundle()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxIconBundle()`

See external documentation.

`new(Ic::wxIconBundle() | wxIcon() (see module wxIcon)) -> wxIconBundle()`

See external documentation.

`new(File::string(), Type::integer()) -> wxIconBundle()`

See external documentation.

`addIcon(This::wxIconBundle(), Icon::wxIcon() (see module wxIcon)) -> ok`

See external documentation.

`addIcon(This::wxIconBundle(), File::string(), Type::integer()) -> ok`

See external documentation.

`getIcon(This::wxIconBundle()) -> wxIcon() (see module wxIcon)`

Equivalent to `getIcon(This, [])`.

`getIcon(This::wxIconBundle(), X::term()) -> wxIcon() (see module wxIcon)`

See external documentation.

Alternatives:

`getIcon(This::wxIconBundle(), [Option]) -> wxIcon:wxIcon()`

Option = {size, integer()}

`getIcon(This::wxIconBundle(), Size::{W::integer(),H::integer()}) -> wxIcon:wxIcon()`

`destroy(This::wxIconBundle()) -> ok`

Destroys this object, do not use object again

## wxIconizeEvent

---

Erlang module

See external documentation: **wxIconizeEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*iconize*

See also the message variant *#wxIconize{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxIconizeEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

`iconized(This::wxIconizeEvent()) -> bool()`

See **external documentation**.



# wxIdleEvent

Erlang module

See external documentation: **wxIdleEvent**.

Use *wxEvtHandler::connect/3* with EventType:

*idle*

See also the message variant *#wxIdle{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

`wxIdleEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`canSend(Win::wxWindow() (see module wxWindow)) -> bool()`

See external documentation.

`getMode() -> WxIdleMode`

Types:

**WxIdleMode** = integer()

See external documentation.

WxIdleMode is one of ?wxIDLE\_PROCESS\_ALL | ?wxIDLE\_PROCESS\_SPECIFIED

`requestMore(This::wxIdleEvent()) -> ok`

Equivalent to *requestMore(This, [])*.

`requestMore(This::wxIdleEvent(), Options::[Option]) -> ok`

Types:

**Option** = {needMore, bool()}

See external documentation.

`moreRequested(This::wxIdleEvent()) -> bool()`

See external documentation.

`setMode(Mode::WxIdleMode) -> ok`

Types:

**WxIdleMode** = integer()

See external documentation.

WxIdleMode is one of ?wxIDLE\_PROCESS\_ALL | ?wxIDLE\_PROCESS\_SPECIFIED

## wxImage

---

Erlang module

See external documentation: **wxImage**.

All (default) image handlers are initialized.

## DATA TYPES

`wxImage()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxImage()`

See **external documentation**.

`new(Name::string()) -> wxImage()`

Equivalent to `new(Name, [])`.

`new(X::integer() | string(), X::integer() | term()) -> wxImage()`

See **external documentation**.

Alternatives:

`new(Width::integer(), Height::integer()) -> new(Width, Height, [])`

`new(Name::string(), [Option]) -> wxImage()`

Option = {type, integer()} | {index, integer()}

`new(X::integer() | string(), X::integer() | string(), X::binary() | term()) -> wxImage()`

See **external documentation**.

Alternatives:

`new(Width::integer(), Height::integer(), Data::binary()) -> new(Width, Height, Data, [])`

`new(Width::integer(), Height::integer(), [Option]) -> wxImage()`

Option = {clear, bool()}

`new(Name::string(), Mimetype::string(), [Option]) -> wxImage()`

Option = {index, integer()}

`new(Width::integer(), Height::integer(), Data::binary(), X::binary() | term()) -> wxImage()`

See **external documentation**.

Alternatives:

```
new(Width::integer(), Height::integer(), Data::binary(), Alpha::binary()) ->  
new(Width,Height,Data,Alpha, [])
```

```
new(Width::integer(), Height::integer(), Data::binary(), [Option]) ->  
wxImage()  
Option = {static_data, bool()}
```

```
new(Width::integer(), Height::integer(), Data::binary(), Alpha::binary(),  
Options::[Option]) -> wxImage()
```

Types:

**Option** = {static\_data, bool()}

See [external documentation](#).

```
blur(This::wxImage(), Radius::integer()) -> wxImage()
```

See [external documentation](#).

```
blurHorizontal(This::wxImage(), Radius::integer()) -> wxImage()
```

See [external documentation](#).

```
blurVertical(This::wxImage(), Radius::integer()) -> wxImage()
```

See [external documentation](#).

```
convertAlphaToMask(This::wxImage()) -> bool()
```

Equivalent to *convertAlphaToMask(This, [])*.

```
convertAlphaToMask(This::wxImage(), Options::[Option]) -> bool()
```

Types:

**Option** = {threshold, integer()}

See [external documentation](#).

```
convertToGreyscale(This::wxImage()) -> wxImage()
```

Equivalent to *convertToGreyscale(This, [])*.

```
convertToGreyscale(This::wxImage(), Options::[Option]) -> wxImage()
```

Types:

**Option** = {lr, float()} | {lg, float()} | {lb, float()}

See [external documentation](#).

```
convertToMono(This::wxImage(), R::integer(), G::integer(), B::integer()) ->  
wxImage()
```

See [external documentation](#).

```
copy(This::wxImage()) -> wxImage()
```

See [external documentation](#).

```
create(This::wxImage(), Width::integer(), Height::integer()) -> bool()
```

Equivalent to *create(This, Width, Height, [])*.

```
create(This::wxImage(), Width::integer(), Height::integer(), X::binary() |  
term()) -> bool()
```

See **external documentation**.

Alternatives:

```
create(This::wxImage(), Width::integer(), Height::integer(), Data::binary())  
-> create(This,Width,Height,Data, [])
```

```
create(This::wxImage(), Width::integer(), Height::integer(), [Option]) ->  
bool()
```

Option = {clear, bool()}

```
create(This::wxImage(), Width::integer(), Height::integer(), Data::binary(),  
X::binary() | term()) -> bool()
```

See **external documentation**.

Alternatives:

```
create(This::wxImage(), Width::integer(), Height::integer(), Data::binary(),  
Alpha::binary()) -> create(This,Width,Height,Data,Alpha, [])
```

```
create(This::wxImage(), Width::integer(), Height::integer(), Data::binary(),  
[Option]) -> bool()
```

Option = {static\_data, bool()}

```
create(This::wxImage(), Width::integer(), Height::integer(), Data::binary(),  
Alpha::binary(), Options::[Option]) -> bool()
```

Types:

**Option = {static\_data, bool()}**

See **external documentation**.

```
Destroy(This::wxImage()) -> ok
```

See **external documentation**.

```
findFirstUnusedColour(This::wxImage()) -> {bool(), R::integer(),  
G::integer(), B::integer()}
```

Equivalent to *findFirstUnusedColour(This, [])*.

```
findFirstUnusedColour(This::wxImage(), Options::[Option]) -> {bool(),  
R::integer(), G::integer(), B::integer()}
```

Types:

**Option = {startR, integer()} | {startG, integer()} | {startB, integer()}**

See **external documentation**.

```
getImageExtWildcard() -> string()
```

See **external documentation**.

`getAlpha(This::wxImage()) -> binary()`

See external documentation.

`getAlpha(This::wxImage(), X::integer(), Y::integer()) -> integer()`

See external documentation.

`getBlue(This::wxImage(), X::integer(), Y::integer()) -> integer()`

See external documentation.

`getData(This::wxImage()) -> binary()`

See external documentation.

`getGreen(This::wxImage(), X::integer(), Y::integer()) -> integer()`

See external documentation.

`getImageCount(Name::string()) -> integer()`

Equivalent to `getImageCount(Name, [])`.

`getImageCount(Name::string(), Options::[Option]) -> integer()`

Types:

`Option = {type, integer()}`

See external documentation.

`getHeight(This::wxImage()) -> integer()`

See external documentation.

`getMaskBlue(This::wxImage()) -> integer()`

See external documentation.

`getMaskGreen(This::wxImage()) -> integer()`

See external documentation.

`getMaskRed(This::wxImage()) -> integer()`

See external documentation.

`getOrFindMaskColour(This::wxImage()) -> {bool(), R::integer(), G::integer(), B::integer()}`

See external documentation.

`getPalette(This::wxImage()) -> wxPalette() (see module wxPalette)`

See external documentation.

`getRed(This::wxImage(), X::integer(), Y::integer()) -> integer()`

See external documentation.

`getSubImage(This::wxImage(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> wxImage()`

See external documentation.

`getWidth(This::wxImage()) -> integer()`

See external documentation.

`hasAlpha(This::wxImage()) -> bool()`

See external documentation.

`hasMask(This::wxImage()) -> bool()`

See external documentation.

`getOption(This::wxImage(), Name::string()) -> string()`

See external documentation.

`getOptionInt(This::wxImage(), Name::string()) -> integer()`

See external documentation.

`hasOption(This::wxImage(), Name::string()) -> bool()`

See external documentation.

`initAlpha(This::wxImage()) -> ok`

See external documentation.

`initStandardHandlers() -> ok`

See external documentation.

`isTransparent(This::wxImage(), X::integer(), Y::integer()) -> bool()`

Equivalent to *isTransparent(This, X, Y, [])*.

`isTransparent(This::wxImage(), X::integer(), Y::integer(), Options::[Option])  
-> bool()`

Types:

**Option** = {threshold, integer()}

See external documentation.

`loadFile(This::wxImage(), Name::string()) -> bool()`

Equivalent to *loadFile(This, Name, [])*.

```
loadFile(This::wxImage(), Name::string(), Options::[Option]) -> bool()
```

Types:

**Option** = {type, integer()} | {index, integer()}

See external documentation.

```
loadFile(This::wxImage(), Name::string(), Mimetype::string(), Options::  
[Option]) -> bool()
```

Types:

**Option** = {index, integer()}

See external documentation.

```
ok(This::wxImage()) -> bool()
```

See external documentation.

```
removeHandler(Name::string()) -> bool()
```

See external documentation.

```
mirror(This::wxImage()) -> wxImage()
```

Equivalent to *mirror(This, [])*.

```
mirror(This::wxImage(), Options::[Option]) -> wxImage()
```

Types:

**Option** = {horizontally, bool()}

See external documentation.

```
replace(This::wxImage(), R1::integer(), G1::integer(), B1::integer(),  
R2::integer(), G2::integer(), B2::integer()) -> ok
```

See external documentation.

```
rescale(This::wxImage(), Width::integer(), Height::integer()) -> wxImage()
```

Equivalent to *rescale(This, Width, Height, [])*.

```
rescale(This::wxImage(), Width::integer(), Height::integer(), Options::  
[Option]) -> wxImage()
```

Types:

**Option** = {quality, integer()}

See external documentation.

```
resize(This::wxImage(), Size::{W::integer(), H::integer()}, Pos::  
{X::integer(), Y::integer()}) -> wxImage()
```

Equivalent to *resize(This, Size, Pos, [])*.

```
resize(This::wxImage(), Size::{W::integer(), H::integer()}, Pos::{X::integer(), Y::integer()}, Options::[Option]) -> wxImage()
```

Types:

**Option** = {r, integer()} | {g, integer()} | {b, integer()}

See external documentation.

```
rotate(This::wxImage(), Angle::float(), Centre_of_rotation::{X::integer(), Y::integer()}) -> wxImage()
```

Equivalent to *rotate(This, Angle, Centre\_of\_rotation, [])*.

```
rotate(This::wxImage(), Angle::float(), Centre_of_rotation::{X::integer(), Y::integer()}, Options::[Option]) -> wxImage()
```

Types:

**Option** = {interpolating, bool()} | {offset\_after\_rotation, {X::integer(), Y::integer()}}

See external documentation.

```
rotateHue(This::wxImage(), Angle::float()) -> ok
```

See external documentation.

```
rotate90(This::wxImage()) -> wxImage()
```

Equivalent to *rotate90(This, [])*.

```
rotate90(This::wxImage(), Options::[Option]) -> wxImage()
```

Types:

**Option** = {clockwise, bool()}

See external documentation.

```
saveFile(This::wxImage(), Name::string()) -> bool()
```

See external documentation.

```
saveFile(This::wxImage(), Name::string(), X::integer() | string()) -> bool()
```

See external documentation.

Alternatives:

```
saveFile(This::wxImage(), Name::string(), Type::integer()) -> bool()
```

```
saveFile(This::wxImage(), Name::string(), Mimetype::string()) -> bool()
```

```
scale(This::wxImage(), Width::integer(), Height::integer()) -> wxImage()
```

Equivalent to *scale(This, Width, Height, [])*.

```
scale(This::wxImage(), Width::integer(), Height::integer(), Options::[Option]) -> wxImage()
```

Types:

**Option** = {quality, integer()}



See [external documentation](#).

```
size(This::wxImage(), Size::{W::integer(), H::integer()}, Pos::{X::integer(), Y::integer()}) -> wxImage()
```

Equivalent to *size(This, Size, Pos, [])*.

```
size(This::wxImage(), Size::{W::integer(), H::integer()}, Pos::{X::integer(), Y::integer()}, Options::[Option]) -> wxImage()
```

Types:

**Option** = {r, integer()} | {g, integer()} | {b, integer()}

See [external documentation](#).

```
setAlpha(This::wxImage(), Alpha::binary()) -> ok
```

Equivalent to *setAlpha(This, Alpha, [])*.

```
setAlpha(This::wxImage(), Alpha::binary(), Options::[Option]) -> ok
```

Types:

**Option** = {static\_data, bool()}

See [external documentation](#).

```
setAlpha(This::wxImage(), X::integer(), Y::integer(), Alpha::integer()) -> ok
```

See [external documentation](#).

```
setData(This::wxImage(), Data::binary()) -> ok
```

Equivalent to *setData(This, Data, [])*.

```
setData(This::wxImage(), Data::binary(), Options::[Option]) -> ok
```

Types:

**Option** = {static\_data, bool()}

See [external documentation](#).

```
setData(This::wxImage(), Data::binary(), New_width::integer(), New_height::integer()) -> ok
```

Equivalent to *setData(This, Data, New\_width, New\_height, [])*.

```
setData(This::wxImage(), Data::binary(), New_width::integer(), New_height::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {static\_data, bool()}

See [external documentation](#).

```
setMask(This::wxImage()) -> ok
```

Equivalent to *setMask(This, [])*.

**setMask(This::wxImage(), Options::[Option]) -> ok**

Types:

**Option = {mask, bool()}**

See external documentation.

**setMaskColour(This::wxImage(), R::integer(), G::integer(), B::integer()) -> ok**

See external documentation.

**setMaskFromImage(This::wxImage(), Mask::wxImage(), Mr::integer(), Mg::integer(), Mb::integer()) -> bool()**

See external documentation.

**setOption(This::wxImage(), Name::string(), X::integer() | string()) -> ok**

See external documentation.

Alternatives:

**setOption(This::wxImage(), Name::string(), Value::integer()) -> ok**

**setOption(This::wxImage(), Name::string(), Value::string()) -> ok**

**setPalette(This::wxImage(), Palette::wxPalette() (see module wxPalette)) -> ok**

See external documentation.

**setRGB(This::wxImage(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}, R::integer(), G::integer(), B::integer()) -> ok**

See external documentation.

**setRGB(This::wxImage(), X::integer(), Y::integer(), R::integer(), G::integer(), B::integer()) -> ok**

See external documentation.

**destroy(This::wxImage()) -> ok**

Destroys this object, do not use object again

# wxImageList

---

Erlang module

See external documentation: **wxImageList**.

## DATA TYPES

`wxImageList()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**`new()`** -> **`wxImageList()`**

See external documentation.

**`new(Width::integer(), Height::integer())`** -> **`wxImageList()`**

Equivalent to *`new(Width, Height, [])`*.

**`new(Width::integer(), Height::integer(), Options::[Option])`** -> **`wxImageList()`**

Types:

**`Option = {mask, bool()} | {initialCount, integer()}`**

See external documentation.

**`add(This::wxImageList(), Bitmap::wxBitmap() (see module wxBitmap))`** -> **`integer()`**

See external documentation.

**`add(This::wxImageList(), Bitmap::wxBitmap() (see module wxBitmap), X::term())`** -> **`integer()`**

See external documentation.

Alternatives:

**`add(This::wxImageList(), Bitmap::wxBitmap:wxBitmap(), Mask::wxBitmap:wxBitmap())`** -> **`integer()`**

**`add(This::wxImageList(), Bitmap::wxBitmap:wxBitmap(), MaskColour::wx:colour())`** -> **`integer()`**

**`create(This::wxImageList(), Width::integer(), Height::integer())`** -> **`bool()`**

Equivalent to *`create(This, Width, Height, [])`*.

**`create(This::wxImageList(), Width::integer(), Height::integer(), Options::[Option])`** -> **`bool()`**

Types:

**`Option = {mask, bool()} | {initialCount, integer()}`**

See external documentation.

```
draw(This::wxImageList(), Index::integer(), Dc::wxDC() (see module wxDC),  
X::integer(), Y::integer()) -> bool()
```

Equivalent to *draw(This, Index, Dc, X, Y, [])*.

```
draw(This::wxImageList(), Index::integer(), Dc::wxDC() (see module wxDC),  
X::integer(), Y::integer(), Options::[Option]) -> bool()
```

Types:

**Option** = {flags, integer()} | {solidBackground, bool()}

See external documentation.

```
getBitmap(This::wxImageList(), Index::integer()) -> wxBitmap() (see module  
wxBitmap)
```

See external documentation.

```
getIcon(This::wxImageList(), Index::integer()) -> wxIcon() (see module  
wxIcon)
```

See external documentation.

```
getImageCount(This::wxImageList()) -> integer()
```

See external documentation.

```
getSize(This::wxImageList(), Index::integer()) -> {bool(), Width::integer(),  
Height::integer()}
```

See external documentation.

```
remove(This::wxImageList(), Index::integer()) -> bool()
```

See external documentation.

```
removeAll(This::wxImageList()) -> bool()
```

See external documentation.

```
replace(This::wxImageList(), Index::integer(), Bitmap::wxBitmap() (see module  
wxBitmap)) -> bool()
```

See external documentation.

```
replace(This::wxImageList(), Index::integer(), Bitmap::wxBitmap() (see module  
wxBitmap), Mask::wxBitmap() (see module wxBitmap)) -> bool()
```

See external documentation.

```
destroy(This::wxImageList()) -> ok
```

Destroys this object, do not use object again

## wxJoystickEvent

---

Erlang module

See external documentation: **wxJoystickEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*joy\_button\_down, joy\_button\_up, joy\_move, joy\_zmove*

See also the message variant *#wxJoystick{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxJoystickEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`buttonDown(This::wxJoystickEvent()) -> bool()`

Equivalent to *buttonDown(This, [])*.

`buttonDown(This::wxJoystickEvent(), Options::[Option]) -> bool()`

Types:

**Option** = {but, integer()}

See **external documentation**.

`buttonIsDown(This::wxJoystickEvent()) -> bool()`

Equivalent to *buttonIsDown(This, [])*.

`buttonIsDown(This::wxJoystickEvent(), Options::[Option]) -> bool()`

Types:

**Option** = {but, integer()}

See **external documentation**.

`buttonUp(This::wxJoystickEvent()) -> bool()`

Equivalent to *buttonUp(This, [])*.

`buttonUp(This::wxJoystickEvent(), Options::[Option]) -> bool()`

Types:

**Option** = {but, integer()}

See **external documentation**.

`getButtonChange(This::wxJoystickEvent()) -> integer()`

See external documentation.

`getButtonState(This::wxJoystickEvent()) -> integer()`

See external documentation.

`getJoystick(This::wxJoystickEvent()) -> integer()`

See external documentation.

`getPosition(This::wxJoystickEvent()) -> {X::integer(), Y::integer()}`

See external documentation.

`getZPosition(This::wxJoystickEvent()) -> integer()`

See external documentation.

`isButton(This::wxJoystickEvent()) -> bool()`

See external documentation.

`isMove(This::wxJoystickEvent()) -> bool()`

See external documentation.

`isZMove(This::wxJoystickEvent()) -> bool()`

See external documentation.

## wxKeyEvent

---

Erlang module

See external documentation: **wxKeyEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*char, char\_hook, key\_down, key\_up*

See also the message variant *#wxKey{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

`wxKeyEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

`altDown(This::wxKeyEvent()) -> bool()`

See external documentation.

`cmdDown(This::wxKeyEvent()) -> bool()`

See external documentation.

`controlDown(This::wxKeyEvent()) -> bool()`

See external documentation.

`getKeyCode(This::wxKeyEvent()) -> integer()`

See external documentation.

`getModifiers(This::wxKeyEvent()) -> integer()`

See external documentation.

`getPosition(This::wxKeyEvent()) -> {X::integer(), Y::integer()}`

See external documentation.

`getRawKeyCode(This::wxKeyEvent()) -> integer()`

See external documentation.

`getRawKeyFlags(This::wxKeyEvent()) -> integer()`

See external documentation.

**getUnicodeKey(This::wxKeyEvent()) -> integer()**

See external documentation.

**getX(This::wxKeyEvent()) -> integer()**

See external documentation.

**getY(This::wxKeyEvent()) -> integer()**

See external documentation.

**hasModifiers(This::wxKeyEvent()) -> bool()**

See external documentation.

**metaDown(This::wxKeyEvent()) -> bool()**

See external documentation.

**shiftDown(This::wxKeyEvent()) -> bool()**

See external documentation.



# wxLayoutAlgorithm

Erlang module

See external documentation: **wxLayoutAlgorithm**.

## DATA TYPES

`wxLayoutAlgorithm()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxLayoutAlgorithm()`

See external documentation.

`layoutFrame(This::wxLayoutAlgorithm(), Frame::wxFrame() (see module wxFrame)) -> bool()`

Equivalent to `layoutFrame(This, Frame, [])`.

`layoutFrame(This::wxLayoutAlgorithm(), Frame::wxFrame() (see module wxFrame), Options::[Option]) -> bool()`

Types:

**Option** = {mainWindow, wxWindow() (see module wxWindow)}

See external documentation.

`layoutMDIFrame(This::wxLayoutAlgorithm(), Frame::wxMDIParentFrame() (see module wxMDIParentFrame)) -> bool()`

Equivalent to `layoutMDIFrame(This, Frame, [])`.

`layoutMDIFrame(This::wxLayoutAlgorithm(), Frame::wxMDIParentFrame() (see module wxMDIParentFrame), Options::[Option]) -> bool()`

Types:

**Option** = {rect, {X::integer(), Y::integer(), W::integer(), H::integer()}}

See external documentation.

`layoutWindow(This::wxLayoutAlgorithm(), Frame::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `layoutWindow(This, Frame, [])`.

`layoutWindow(This::wxLayoutAlgorithm(), Frame::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {mainWindow, wxWindow() (see module wxWindow)}

## **wxLayoutAlgorithm**

---

See **external documentation**.

**destroy(This::wxLayoutAlgorithm()) -> ok**

Destroys this object, do not use object again

## wxListBox

---

Erlang module

See external documentation: **wxListBox**.

This class is derived (and can use functions) from:

*wxControlWithItems*

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxListBox()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxListBox()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxListBox()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxListBox()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {choices, [[string()]]} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxListBox(), Parent::wxWindow() (see module wxWindow), Id::integer(), Pos::{X::integer(), Y::integer()}, Size::{W::integer(), H::integer()}, Choices::[[string()]]) -> bool()`

Equivalent to `create(This, Parent, Id, Pos, Size, Choices, [])`.

`create(This::wxListBox(), Parent::wxWindow() (see module wxWindow), Id::integer(), Pos::{X::integer(), Y::integer()}, Size::{W::integer(), H::integer()}, Choices::[[string()]], Options::[Option]) -> bool()`

Types:

**Option** = {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

**deselect**(This::wxListBox(), N::integer()) -> ok

See external documentation.

**getSelections**(This::wxListBox()) -> {integer(), ASelections::[integer()]}

See external documentation.

**insertItems**(This::wxListBox(), Items::[[string()]], Pos::integer()) -> ok

See external documentation.

**isSelected**(This::wxListBox(), N::integer()) -> bool()

See external documentation.

**set**(This::wxListBox(), Items::[[string()]]) -> ok

See external documentation.

**hitTest**(This::wxListBox(), Point::{X::integer(), Y::integer()}) -> integer()

See external documentation.

**setFirstItem**(This::wxListBox(), X::integer() | string()) -> ok

See external documentation.

Alternatives:

**setFirstItem**(This::wxListBox(), N::integer()) -> ok

**setFirstItem**(This::wxListBox(), S::string()) -> ok

**destroy**(This::wxListBox()) -> ok

Destroys this object, do not use object again

# wxListCtrl

Erlang module

See external documentation: **wxListCtrl**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxListCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxListCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxListCtrl()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxListCtrl()`

Types:

**Option** = {winid, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`arrange(This::wxListCtrl()) -> bool()`

Equivalent to `arrange(This, [])`.

`arrange(This::wxListCtrl(), Options::[Option]) -> bool()`

Types:

**Option** = {flag, integer()}

See external documentation.

`assignImageList(This::wxListCtrl(), ImageList::wxImageList() (see module wxImageList), Which::integer()) -> ok`

See external documentation.

`clearAll(This::wxListCtrl()) -> ok`

See external documentation.

```
create(This::wxListCtrl(), Parent::wxWindow() (see module wxWindow)) -> bool()
```

Equivalent to *create(This, Parent, [])*.

```
create(This::wxListCtrl(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()
```

Types:

```
Option = {winid, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}
```

See external documentation.

```
deleteAllItems(This::wxListCtrl()) -> bool()
```

See external documentation.

```
deleteColumn(This::wxListCtrl(), Col::integer()) -> bool()
```

See external documentation.

```
deleteItem(This::wxListCtrl(), Item::integer()) -> bool()
```

See external documentation.

```
editLabel(This::wxListCtrl(), Item::integer()) -> wxTextCtrl() (see module wxTextCtrl)
```

See external documentation.

```
ensureVisible(This::wxListCtrl(), Item::integer()) -> bool()
```

See external documentation.

```
findItem(This::wxListCtrl(), Start::integer(), Str::string()) -> integer()
```

Equivalent to *findItem(This, Start, Str, [])*.

```
findItem(This::wxListCtrl(), Start::integer(), X::string() | term(), X::term() | integer()) -> integer()
```

See external documentation.

Alternatives:

```
findItem(This::wxListCtrl(), Start::integer(), Str::string(), [Option]) -> integer()
```

Option = {partial, bool() }

```
findItem(This::wxListCtrl(), Start::integer(), Pt:: {X::integer(), Y::integer()}, Direction::integer()) -> integer()
```

```
getColumn(This::wxListCtrl(), Col::integer(), Item::wxListItem() (see module wxListItem)) -> bool()
```

See external documentation.

`getColumnCount(This::wxListCtrl()) -> integer()`

See external documentation.

`getColumnWidth(This::wxListCtrl(), Col::integer()) -> integer()`

See external documentation.

`getCountPerPage(This::wxListCtrl()) -> integer()`

See external documentation.

`getEditControl(This::wxListCtrl()) -> wxTextCtrl() (see module wxTextCtrl)`

See external documentation.

`getImageList(This::wxListCtrl(), Which::integer()) -> wxImageList() (see module wxImageList)`

See external documentation.

`getItem(This::wxListCtrl(), Info::wxListItem() (see module wxListItem)) -> bool()`

See external documentation.

`getItemBackgroundColour(This::wxListCtrl(), Item::integer()) -> colour() (see module wx)`

See external documentation.

`getItemCount(This::wxListCtrl()) -> integer()`

See external documentation.

`getItemData(This::wxListCtrl(), Item::integer()) -> integer()`

See external documentation.

`getItemFont(This::wxListCtrl(), Item::integer()) -> wxFont() (see module wxFont)`

See external documentation.

`getItemPosition(This::wxListCtrl(), Item::integer(), Pos::{X::integer(), Y::integer()}) -> bool()`

See external documentation.

`getItemRect(This::wxListCtrl(), Item::integer(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> bool()`

Equivalent to `getItemRect(This, Item, Rect, [])`.

```
getItemRect(This::wxListCtrl(), Item::integer(), Rect::{X::integer(),  
Y::integer(), W::integer(), H::integer()}, Options::[Option]) -> bool()
```

Types:

**Option** = {code, integer()}

See external documentation.

```
getItemSpacing(This::wxListCtrl()) -> {W::integer(), H::integer()}
```

See external documentation.

```
getItemState(This::wxListCtrl(), Item::integer(), StateMask::integer()) ->  
integer()
```

See external documentation.

```
getItemText(This::wxListCtrl(), Item::integer()) -> string()
```

See external documentation.

```
getItemTextColour(This::wxListCtrl(), Item::integer()) -> colour() (see  
module wx)
```

See external documentation.

```
getNextItem(This::wxListCtrl(), Item::integer()) -> integer()
```

Equivalent to *getNextItem(This, Item, [])*.

```
getNextItem(This::wxListCtrl(), Item::integer(), Options::[Option]) ->  
integer()
```

Types:

**Option** = {geometry, integer()} | {state, integer()}

See external documentation.

```
getSelectedItemCount(This::wxListCtrl()) -> integer()
```

See external documentation.

```
getTextColour(This::wxListCtrl()) -> colour() (see module wx)
```

See external documentation.

```
getTopItem(This::wxListCtrl()) -> integer()
```

See external documentation.

```
getViewRect(This::wxListCtrl()) -> {X::integer(), Y::integer(), W::integer(),  
H::integer()}
```

See external documentation.



```
hitTest(This::wxListCtrl(), Point::{X::integer(), Y::integer()}) ->
{integer(), Flags::integer()}
```

See external documentation.

```
insertColumn(This::wxListCtrl(), Col::integer(), X::string() | term()) ->
integer()
```

See external documentation.

Alternatives:

```
insertColumn(This::wxListCtrl(), Col::integer(), Heading::string()) ->
insertColumn(This, Col, Heading, [])
```

```
insertColumn(This::wxListCtrl(), Col::integer(),
Info::wxListItem:wxListItem()) -> integer()
```

```
insertColumn(This::wxListCtrl(), Col::integer(), Heading::string(), Options::
[Option]) -> integer()
```

Types:

**Option** = {format, integer()} | {width, integer()}

See external documentation.

```
insertItem(This::wxListCtrl(), Info::wxListItem() (see module wxListItem)) ->
integer()
```

See external documentation.

```
insertItem(This::wxListCtrl(), Index::integer(), X::integer() | string()) ->
integer()
```

See external documentation.

Alternatives:

```
insertItem(This::wxListCtrl(), Index::integer(), ImageIndex::integer()) ->
integer()
```

```
insertItem(This::wxListCtrl(), Index::integer(), Label::string()) -> integer()
```

```
insertItem(This::wxListCtrl(), Index::integer(), Label::string(),
ImageIndex::integer()) -> integer()
```

See external documentation.

```
refreshItem(This::wxListCtrl(), Item::integer()) -> ok
```

See external documentation.

```
refreshItems(This::wxListCtrl(), ItemFrom::integer(), ItemTo::integer()) ->
ok
```

See external documentation.

```
scrollList(This::wxListCtrl(), Dx::integer(), Dy::integer()) -> bool()
```

See external documentation.

`setBackgroundColour(This::wxListCtrl(), Colour::colour() (see module wx)) -> bool()`

See external documentation.

`setColumn(This::wxListCtrl(), Col::integer(), Item::wxListItem() (see module wxListItem)) -> bool()`

See external documentation.

`setColumnWidth(This::wxListCtrl(), Col::integer(), Width::integer()) -> bool()`

See external documentation.

`setImageList(This::wxListCtrl(), ImageList::wxImageList() (see module wxImageList), Which::integer()) -> ok`

See external documentation.

`setItem(This::wxListCtrl(), Info::wxListItem() (see module wxListItem)) -> bool()`

See external documentation.

`setItem(This::wxListCtrl(), Index::integer(), Col::integer(), Label::string()) -> integer()`

Equivalent to *setItem(This, Index, Col, Label, [])*.

`setItem(This::wxListCtrl(), Index::integer(), Col::integer(), Label::string(), Options::[Option]) -> integer()`

Types:

`Option = {imageId, integer()}`

See external documentation.

`setItemBackgroundColour(This::wxListCtrl(), Item::integer(), Col::colour() (see module wx)) -> ok`

See external documentation.

`setItemCount(This::wxListCtrl(), Count::integer()) -> ok`

See external documentation.

`setItemData(This::wxListCtrl(), Item::integer(), Data::integer()) -> bool()`

See external documentation.

`setItemFont(This::wxListCtrl(), Item::integer(), F::wxFont() (see module wxFont)) -> ok`

See external documentation.

```
setItemImage(This::wxListCtrl(), Item::integer(), Image::integer()) -> bool()
```

Equivalent to *setItemImage(This, Item, Image, [])*.

```
setItemImage(This::wxListCtrl(), Item::integer(), Image::integer(), Options::[Option]) -> bool()
```

Types:

**Option** = {setImage, integer()}

See external documentation.

```
setItemColumnImage(This::wxListCtrl(), Item::integer(), Column::integer(), Image::integer()) -> bool()
```

See external documentation.

```
setItemPosition(This::wxListCtrl(), Item::integer(), Pos::{X::integer(), Y::integer()}) -> bool()
```

See external documentation.

```
setItemState(This::wxListCtrl(), Item::integer(), State::integer(), StateMask::integer()) -> bool()
```

See external documentation.

```
setItemText(This::wxListCtrl(), Item::integer(), Str::string()) -> ok
```

See external documentation.

```
setItemTextColour(This::wxListCtrl(), Item::integer(), Col::colour() (see module wx)) -> ok
```

See external documentation.

```
setSingleStyle(This::wxListCtrl(), Style::integer()) -> ok
```

Equivalent to *setSingleStyle(This, Style, [])*.

```
setSingleStyle(This::wxListCtrl(), Style::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {add, bool()}

See external documentation.

```
setTextColour(This::wxListCtrl(), Col::colour() (see module wx)) -> ok
```

See external documentation.

```
setWindowStyleFlag(This::wxListCtrl(), Style::integer()) -> ok
```

See external documentation.

## **wxListCtrl**

---

**sortItems(This::wxListCtrl(), SortCallback::function()) -> boolean()**

Sort the items in the list control

```
SortCalBack(Item1,Item2) -> integer()
```

SortCalBack receives the client data associated with two items to compare, and should return 0 if the items are equal, a negative value if the first item is less than the second one and a positive value if the first item is greater than the second one.

NOTE: The callback may not call other processes.

**destroy(This::wxListCtrl()) -> ok**

Destroys this object, do not use object again

## wxListEvent

Erlang module

See external documentation: **wxListEvent**.

Use *wxEvtHandler:connect/3* with EventType:

<i>command_list_begin_drag,</i>	<i>command_list_begin_rdrag,</i>	<i>command_list_begin_label_edit,</i>
<i>command_list_end_label_edit,</i>	<i>command_list_delete_item,</i>	<i>command_list_delete_all_items,</i>
<i>command_list_key_down,</i>	<i>command_list_insert_item,</i>	<i>command_list_col_click,</i>
<i>command_list_col_right_click,</i>		
<i>command_list_col_begin_drag,</i>	<i>command_list_col_dragging,</i>	<i>command_list_col_end_drag,</i>
<i>command_list_item_selected,</i>	<i>command_list_item_deselected,</i>	<i>command_list_item_right_click,</i>
<i>command_list_item_middle_click,</i>	<i>command_list_item_activated,</i>	<i>command_list_item_focused,</i>
<i>command_list_cache_hint</i>		

See also the message variant *#wxList{}* event record type.

This class is derived (and can use functions) from:

*wxNotifyEvent*  
*wxCommandEvent*  
*wxEvt*

## DATA TYPES

`wxListEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

`getCacheFrom(This::wxListEvent()) -> integer()`

See external documentation.

`getCacheTo(This::wxListEvent()) -> integer()`

See external documentation.

`getKeyCode(This::wxListEvent()) -> integer()`

See external documentation.

`getIndex(This::wxListEvent()) -> integer()`

See external documentation.

`getColumn(This::wxListEvent()) -> integer()`

See external documentation.

`getPoint(This::wxListEvent()) -> {X::integer(), Y::integer()}`

See external documentation.

`getLabel(This::wxListEvent()) -> string()`

See external documentation.

`getText(This::wxListEvent()) -> string()`

See external documentation.

`getImage(This::wxListEvent()) -> integer()`

See external documentation.

`getData(This::wxListEvent()) -> integer()`

See external documentation.

`getMask(This::wxListEvent()) -> integer()`

See external documentation.

`getItem(This::wxListEvent()) -> wxListItem() (see module wxListItem)`

See external documentation.

`isEditCancelled(This::wxListEvent()) -> bool()`

See external documentation.

## wxListItem

---

Erlang module

See external documentation: **wxListItem**.

### DATA TYPES

`wxListItem()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxListItem()`

See external documentation.

`new(Item::wxListItem()) -> wxListItem()`

See external documentation.

`clear(This::wxListItem()) -> ok`

See external documentation.

`getAlign(This::wxListItem()) -> WxListColumnFormat`

Types:

`WxListColumnFormat = integer()`

See external documentation.

`WxListColumnFormat` is one of `?wxLIST_FORMAT_LEFT` | `?wxLIST_FORMAT_RIGHT` | `?wxLIST_FORMAT_CENTRE` | `?wxLIST_FORMAT_CENTER`

`getBackgroundColour(This::wxListItem()) -> colour() (see module wx)`

See external documentation.

`getColumn(This::wxListItem()) -> integer()`

See external documentation.

`getFont(This::wxListItem()) -> wxFont() (see module wxFont)`

See external documentation.

`getId(This::wxListItem()) -> integer()`

See external documentation.

`getImage(This::wxListItem()) -> integer()`

See external documentation.

`getMask(This::wxListItem()) -> integer()`

See external documentation.

`getState(This::wxListItem()) -> integer()`

See external documentation.

`getText(This::wxListItem()) -> string()`

See external documentation.

`getTextColour(This::wxListItem()) -> colour() (see module wx)`

See external documentation.

`getWidth(This::wxListItem()) -> integer()`

See external documentation.

`setAlign(This::wxListItem(), Align::WxListColumnFormat) -> ok`

Types:

`WxListColumnFormat = integer()`

See external documentation.

WxListColumnFormat is one of ?wxLIST\_FORMAT\_LEFT | ?wxLIST\_FORMAT\_RIGHT | ?wxLIST\_FORMAT\_CENTRE | ?wxLIST\_FORMAT\_CENTER

`setBackgroundColour(This::wxListItem(), ColBack::colour() (see module wx)) -> ok`

See external documentation.

`setColumn(This::wxListItem(), Col::integer()) -> ok`

See external documentation.

`setFont(This::wxListItem(), Font::wxFont() (see module wxFont)) -> ok`

See external documentation.

`setId(This::wxListItem(), Id::integer()) -> ok`

See external documentation.

`setImage(This::wxListItem(), Image::integer()) -> ok`

See external documentation.

`setMask(This::wxListItem(), Mask::integer()) -> ok`

See external documentation.

`setState(This::wxListItem(), State::integer()) -> ok`

See external documentation.



**setStateMask(This::wxListItem(), StateMask::integer()) -> ok**

See external documentation.

**setText(This::wxListItem(), Text::string()) -> ok**

See external documentation.

**setTextColour(This::wxListItem(), ColText::colour() (see module wx)) -> ok**

See external documentation.

**setWidth(This::wxListItem(), Width::integer()) -> ok**

See external documentation.

**destroy(This::wxListItem()) -> ok**

Destroys this object, do not use object again

## wxListView

---

Erlang module

See external documentation: **wxListView**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxListView()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`clearColumnImage(This::wxListView(), Col::integer()) -> ok`

See external documentation.

`focus(This::wxListView(), Index::integer()) -> ok`

See external documentation.

`getFirstSelected(This::wxListView()) -> integer()`

See external documentation.

`getFocusedItem(This::wxListView()) -> integer()`

See external documentation.

`getNextSelected(This::wxListView(), Item::integer()) -> integer()`

See external documentation.

`isSelected(This::wxListView(), Index::integer()) -> bool()`

See external documentation.

`select(This::wxListView(), N::integer()) -> ok`

Equivalent to *select(This, N, [])*.

`select(This::wxListView(), N::integer(), Options::[Option]) -> ok`

Types:

**Option** = {on, bool()}

See external documentation.

`setColumnImage(This::wxListView(), Col::integer(), Image::integer()) -> ok`

See external documentation.

## wxListbook

---

Erlang module

See external documentation: **wxListbook**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxListbook()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxListbook()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxListbook()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxListbook()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`addPage(This::wxListbook(), Page::wxWindow() (see module wxWindow), Text::string()) -> bool()`

Equivalent to `addPage(This, Page, Text, [])`.

`addPage(This::wxListbook(), Page::wxWindow() (see module wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

**Option** = {bSelect, bool()} | {imageId, integer()}

See external documentation.

`advanceSelection(This::wxListbook()) -> ok`

Equivalent to `advanceSelection(This, [])`.

`advanceSelection(This::wxListbook(), Options::[Option]) -> ok`

Types:

**Option** = {forward, bool()}

See external documentation.

**assignImageList**(This::wxListbook(), ImageList::wxImageList() (see module wxImageList)) -> ok

See external documentation.

**create**(This::wxListbook(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()

Equivalent to *create(This, Parent, Id, [])*.

**create**(This::wxListbook(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

**deleteAllPages**(This::wxListbook()) -> bool()

See external documentation.

**deletePage**(This::wxListbook(), N::integer()) -> bool()

See external documentation.

**removePage**(This::wxListbook(), N::integer()) -> bool()

See external documentation.

**getCurrentPage**(This::wxListbook()) -> wxWindow() (see module wxWindow)

See external documentation.

**getImageList**(This::wxListbook()) -> wxImageList() (see module wxImageList)

See external documentation.

**getPage**(This::wxListbook(), N::integer()) -> wxWindow() (see module wxWindow)

See external documentation.

**getPageCount**(This::wxListbook()) -> integer()

See external documentation.

**getPageImage**(This::wxListbook(), N::integer()) -> integer()

See external documentation.

**getPageText**(This::wxListbook(), N::integer()) -> string()

See external documentation.

`getSelection(This::wxListbook()) -> integer()`

See external documentation.

`hitTest(This::wxListbook(), Pt::{X::integer(), Y::integer()}) -> {integer(),  
Flags::integer()}`

See external documentation.

`insertPage(This::wxListbook(), N::integer(), Page::wxWindow() (see module  
wxWindow), Text::string()) -> bool()`

Equivalent to *insertPage(This, N, Page, Text, [])*.

`insertPage(This::wxListbook(), N::integer(), Page::wxWindow() (see module  
wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

`Option = {bSelect, bool()} | {imageId, integer()}`

See external documentation.

`setImageList(This::wxListbook(), ImageList::wxImageList() (see module  
wxImageList)) -> ok`

See external documentation.

`setPageSize(This::wxListbook(), Size::{W::integer(), H::integer()}) -> ok`

See external documentation.

`setPageImage(This::wxListbook(), N::integer(), ImageId::integer()) -> bool()`

See external documentation.

`setPageText(This::wxListbook(), N::integer(), StrText::string()) -> bool()`

See external documentation.

`setSelection(This::wxListbook(), N::integer()) -> integer()`

See external documentation.

`changeSelection(This::wxListbook(), N::integer()) -> integer()`

See external documentation.

`destroy(This::wxListbook()) -> ok`

Destroys this object, do not use object again

## wxLogNull

---

Erlang module

See external documentation: **wxLogNull**.

### DATA TYPES

`wxLogNull()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxLogNull()`**

See external documentation.

**`destroy(This::wxLogNull())`** -> **`ok`**

Destroys this object, do not use object again

## wxMDIChildFrame

---

Erlang module

See external documentation: **wxMDIChildFrame**.

This class is derived (and can use functions) from:

*wxFrame*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxMDIChildFrame()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxMDIChildFrame()`

See external documentation.

`new(Parent::wxMDIParentFrame() (see module wxMDIParentFrame), Id::integer(), Title::string()) -> wxMDIChildFrame()`

Equivalent to `new(Parent, Id, Title, [])`.

`new(Parent::wxMDIParentFrame() (see module wxMDIParentFrame), Id::integer(), Title::string(), Options::[Option]) -> wxMDIChildFrame()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`activate(This::wxMDIChildFrame()) -> ok`

See external documentation.

`create(This::wxMDIChildFrame(), Parent::wxMDIParentFrame() (see module wxMDIParentFrame), Id::integer(), Title::string()) -> bool()`

Equivalent to `create(This, Parent, Id, Title, [])`.

`create(This::wxMDIChildFrame(), Parent::wxMDIParentFrame() (see module wxMDIParentFrame), Id::integer(), Title::string(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.



**maximize(This::wxMDIChildFrame()) -> ok**

Equivalent to *maximize(This, [])*.

**maximize(This::wxMDIChildFrame(), Options::[Option]) -> ok**

Types:

**Option = {maximize, bool()}**

See **external documentation**.

**restore(This::wxMDIChildFrame()) -> ok**

See **external documentation**.

**destroy(This::wxMDIChildFrame()) -> ok**

Destroys this object, do not use object again

## wxMDIClientWindow

---

Erlang module

See external documentation: **wxMDIClientWindow**.

This class is derived (and can use functions) from:

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxMDIClientWindow()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxMDIClientWindow()`

See external documentation.

`new(Parent::wxMDIParentFrame() (see module wxMDIParentFrame)) -> wxMDIClientWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxMDIParentFrame() (see module wxMDIParentFrame), Options::[Option]) -> wxMDIClientWindow()`

Types:

**Option** = {style, integer()}

See external documentation.

`createClient(This::wxMDIClientWindow(), Parent::wxMDIParentFrame() (see module wxMDIParentFrame)) -> bool()`

Equivalent to `createClient(This, Parent, [])`.

`createClient(This::wxMDIClientWindow(), Parent::wxMDIParentFrame() (see module wxMDIParentFrame), Options::[Option]) -> bool()`

Types:

**Option** = {style, integer()}

See external documentation.

`destroy(This::wxMDIClientWindow()) -> ok`

Destroys this object, do not use object again

# wxMDIParentFrame

Erlang module

See external documentation: **wxMDIParentFrame**.

This class is derived (and can use functions) from:

*wxFrame*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxMDIParentFrame()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxMDIParentFrame()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string()) -> wxMDIParentFrame()`

Equivalent to `new(Parent, Id, Title, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string(), Options::[Option]) -> wxMDIParentFrame()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`activateNext(This::wxMDIParentFrame()) -> ok`

See external documentation.

`activatePrevious(This::wxMDIParentFrame()) -> ok`

See external documentation.

`arrangeIcons(This::wxMDIParentFrame()) -> ok`

See external documentation.

`cascade(This::wxMDIParentFrame()) -> ok`

See external documentation.

```
create(This::wxMDIParentFrame(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Title::string()) -> bool()
```

Equivalent to *create(This, Parent, Id, Title, [])*.

```
create(This::wxMDIParentFrame(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Title::string(), Options::[Option]) -> bool()
```

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See [external documentation](#).

```
getActiveChild(This::wxMDIParentFrame()) -> wxMDIChildFrame() (see module  
wxMDIChildFrame)
```

See [external documentation](#).

```
getClientWindow(This::wxMDIParentFrame()) -> wxMDIClientWindow() (see module  
wxMDIClientWindow)
```

See [external documentation](#).

```
tile(This::wxMDIParentFrame()) -> ok
```

Equivalent to *tile(This, [])*.

```
tile(This::wxMDIParentFrame(), Options::[Option]) -> ok
```

Types:

**Option** = {orient, WxOrientation}

**WxOrientation** = integer()

See [external documentation](#).

WxOrientation is one of ?wxHORIZONTAL | ?wxVERTICAL | ?wxBOTH

```
destroy(This::wxMDIParentFrame()) -> ok
```

Destroys this object, do not use object again

# wxMask

Erlang module

See external documentation: **wxMask**.

## DATA TYPES

`wxMask()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxMask()`

See external documentation.

`new(Bitmap::wxBitmap()) (see module wxBitmap) -> wxMask()`

See external documentation.

`new(Bitmap::wxBitmap() (see module wxBitmap), X::integer() | term()) -> wxMask()`

See external documentation.

Alternatives:

`new(Bitmap::wxBitmap:wxBitmap(), PaletteIndex::integer()) -> wxMask()`

`new(Bitmap::wxBitmap:wxBitmap(), Colour::wx:colour()) -> wxMask()`

`create(This::wxMask(), Bitmap::wxBitmap() (see module wxBitmap)) -> bool()`

See external documentation.

`create(This::wxMask(), Bitmap::wxBitmap() (see module wxBitmap), X::integer() | term()) -> bool()`

See external documentation.

Alternatives:

`create(This::wxMask(), Bitmap::wxBitmap:wxBitmap(), PaletteIndex::integer()) -> bool()`

`create(This::wxMask(), Bitmap::wxBitmap:wxBitmap(), Colour::wx:colour()) -> bool()`

`destroy(This::wxMask()) -> ok`

Destroys this object, do not use object again

## wxMaximizeEvent

---

Erlang module

See external documentation: **wxMaximizeEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*maximize*

See also the message variant *#wxMaximize{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxMaximizeEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

# wxMemoryDC

---

Erlang module

See external documentation: **wxMemoryDC**.

This class is derived (and can use functions) from:  
*wxDC*

## DATA TYPES

`wxMemoryDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxMemoryDC()`

See external documentation.

`new(Dc::wxDC() (see module wxDC) | wxBitmap() (see module wxBitmap)) -> wxMemoryDC()`

See external documentation.

`selectObject(This::wxMemoryDC(), Bmp::wxBitmap() (see module wxBitmap)) -> ok`

See external documentation.

`selectObjectAsSource(This::wxMemoryDC(), Bmp::wxBitmap() (see module wxBitmap)) -> ok`

See external documentation.

`destroy(This::wxMemoryDC()) -> ok`

Destroys this object, do not use object again

## wxMenu

---

Erlang module

See external documentation: **wxMenu**.

This class is derived (and can use functions) from:  
*wxEvtHandler*

## DATA TYPES

`wxMenu()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxMenu()`

Equivalent to `new([])`.

`new(Options::[Option]) -> wxMenu()`

Types:

**Option** = {style, integer()}

See external documentation.

`new(Title::string(), Options::[Option]) -> wxMenu()`

Types:

**Option** = {style, integer()}

See external documentation.

`append(This::wxMenu(), Item::wxMenuItem() (see module wxMenuItem)) -> wxMenuItem() (see module wxMenuItem)`

See external documentation.

`append(This::wxMenu(), Itemid::integer(), Text::string()) -> wxMenuItem() (see module wxMenuItem)`

Equivalent to `append(This, Itemid, Text, [])`.

`append(This::wxMenu(), Itemid::integer(), Text::string(), X::wxMenu() | term()) -> wxMenuItem() (see module wxMenuItem)`

See external documentation.

Alternatives:

`append(This::wxMenu(), Itemid::integer(), Text::string(), Submenu::wxMenu()) -> append(This, Itemid, Text, Submenu, [])`

`append(This::wxMenu(), Itemid::integer(), Text::string(), [Option]) -> wxMenuItem:wxMenuItem()`



Option = {help, string()} | {kind, WxItemKind}

WxItemKind = integer()

WxItemKind is one of ?wxITEM\_SEPARATOR | ?wxITEM\_NORMAL | ?wxITEM\_CHECK | ?wxITEM\_RADIO | ?wxITEM\_MAX

**append(This::wxMenu(), Itemid::integer(), Text::string(), X::string() | wxMenu(), X::bool() | term()) -> ok | wxMenuItem() (see module wxMenuItem)**

See **external documentation**.

Alternatives:

**append(This::wxMenu(), Itemid::integer(), Text::string(), Help::string(), IsCheckable::bool()) -> ok**

**append(This::wxMenu(), Itemid::integer(), Text::string(), Submenu::wxMenu(), [Option]) -> wxMenuItem:wxMenuItem()**

Option = {help, string() }

**appendCheckItem(This::wxMenu(), Itemid::integer(), Text::string()) -> wxMenuItem() (see module wxMenuItem)**

Equivalent to *appendCheckItem(This, Itemid, Text, [])*.

**appendCheckItem(This::wxMenu(), Itemid::integer(), Text::string(), Options:: [Option]) -> wxMenuItem() (see module wxMenuItem)**

Types:

**Option = {help, string() }**

See **external documentation**.

**appendRadioItem(This::wxMenu(), Itemid::integer(), Text::string()) -> wxMenuItem() (see module wxMenuItem)**

Equivalent to *appendRadioItem(This, Itemid, Text, [])*.

**appendRadioItem(This::wxMenu(), Itemid::integer(), Text::string(), Options:: [Option]) -> wxMenuItem() (see module wxMenuItem)**

Types:

**Option = {help, string() }**

See **external documentation**.

**appendSeparator(This::wxMenu()) -> wxMenuItem() (see module wxMenuItem)**

See **external documentation**.

**break(This::wxMenu()) -> ok**

See **external documentation**.

**check(This::wxMenu(), Itemid::integer(), Check::bool()) -> ok**

See **external documentation**.

**delete(This::wxMenu(), X::integer() | term()) -> bool()**

See [external documentation](#).

Alternatives:

**delete(This::wxMenu(), Itemid::integer()) -> bool()**

**delete(This::wxMenu(), Item::wxMenuItem:wxMenuItem()) -> bool()**

**Destroy(This::wxMenu(), X::integer() | term()) -> bool()**

See [external documentation](#).

Alternatives:

**'Destroy'(This::wxMenu(), Itemid::integer()) -> bool()**

**'Destroy'(This::wxMenu(), Item::wxMenuItem:wxMenuItem()) -> bool()**

**enable(This::wxMenu(), Itemid::integer(), Enable::bool()) -> ok**

See [external documentation](#).

**findItem(This::wxMenu(), X::integer() | string()) -> wxMenuItem() (see module wxMenuItem) | integer()**

See [external documentation](#).

Alternatives:

**findItem(This::wxMenu(), Itemid::integer()) -> wxMenuItem:wxMenuItem()**

**findItem(This::wxMenu(), Item::string()) -> integer()**

**findItemByPosition(This::wxMenu(), Position::integer()) -> wxMenuItem() (see module wxMenuItem)**

See [external documentation](#).

**getHelpString(This::wxMenu(), Itemid::integer()) -> string()**

See [external documentation](#).

**getLabel(This::wxMenu(), Itemid::integer()) -> string()**

See [external documentation](#).

**getMenuItemCount(This::wxMenu()) -> integer()**

See [external documentation](#).

**getMenuItems(This::wxMenu()) -> [wxMenuItem() (see module wxMenuItem)]**

See [external documentation](#).

**getTitle(This::wxMenu()) -> string()**

See [external documentation](#).

---

```
insert(This::wxMenu(), Pos::integer(), X::integer() | term()) -> wxMenuItem()
(see module wxMenuItem)
```

See external documentation.

Alternatives:

```
insert(This::wxMenu(),      Pos::integer(),      Itemid::integer())      ->
insert(This, Pos, Itemid, [])

insert(This::wxMenu(),  Pos::integer(),  Item::wxMenuItem:wxMenuItem()) ->
wxMenuItem:wxMenuItem()
```

```
insert(This::wxMenu(), Pos::integer(), Itemid::integer(), Options::[Option])
-> wxMenuItem() (see module wxMenuItem)
```

Types:

**Option** = {text, string()} | {help, string()} | {kind, WxItemKind}

**WxItemKind** = integer()

See external documentation.

WxItemKind is one of ?wxITEM\_SEPARATOR | ?wxITEM\_NORMAL | ?wxITEM\_CHECK | ?wxITEM\_RADIO  
| ?wxITEM\_MAX

```
insert(This::wxMenu(), Pos::integer(), Itemid::integer(), Text::string(),
Submenu::wxMenu()) -> wxMenuItem() (see module wxMenuItem)
```

Equivalent to *insert(This, Pos, Itemid, Text, Submenu, [])*.

```
insert(This::wxMenu(), Pos::integer(), Itemid::integer(), Text::string(),
X::string() | wxMenu(), X::bool() | term()) -> ok | wxMenuItem() (see module
wxMenuItem)
```

See external documentation.

Alternatives:

```
insert(This::wxMenu(),  Pos::integer(),  Itemid::integer(),  Text::string(),
Help::string(), IsCheckable::bool()) -> ok

insert(This::wxMenu(),  Pos::integer(),  Itemid::integer(),  Text::string(),
Submenu::wxMenu(), [Option]) -> wxMenuItem:wxMenuItem()
Option = {help, string()}
```

```
insertCheckItem(This::wxMenu(), Pos::integer(), Itemid::integer(),
Text::string()) -> wxMenuItem() (see module wxMenuItem)
```

Equivalent to *insertCheckItem(This, Pos, Itemid, Text, [])*.

```
insertCheckItem(This::wxMenu(), Pos::integer(), Itemid::integer(),
Text::string(), Options::[Option]) -> wxMenuItem() (see module wxMenuItem)
```

Types:

**Option** = {help, string()}

See external documentation.

```
insertRadioItem(This::wxMenu(), Pos::integer(), Itemid::integer(),
Text::string()) -> wxMenuItem() (see module wxMenuItem)
```

Equivalent to *insertRadioItem(This, Pos, Itemid, Text, [])*.

```
insertRadioItem(This::wxMenu(), Pos::integer(), Itemid::integer(),
Text::string(), Options::[Option]) -> wxMenuItem() (see module wxMenuItem)
```

Types:

**Option** = {help, string() }

See [external documentation](#).

```
insertSeparator(This::wxMenu(), Pos::integer()) -> wxMenuItem() (see module
wxMenuItem)
```

See [external documentation](#).

```
isChecked(This::wxMenu(), Itemid::integer()) -> bool()
```

See [external documentation](#).

```
isEnabled(This::wxMenu(), Itemid::integer()) -> bool()
```

See [external documentation](#).

```
prepend(This::wxMenu(), X::integer() | term()) -> wxMenuItem() (see module
wxMenuItem)
```

See [external documentation](#).

Alternatives:

```
prepend(This::wxMenu(), Itemid::integer()) -> prepend(This, Itemid, [])
```

```
prepend(This::wxMenu(), Item::wxMenuItem:wxMenuItem()) ->
wxMenuItem:wxMenuItem()
```

```
prepend(This::wxMenu(), Itemid::integer(), Options::[Option]) -> wxMenuItem()
(see module wxMenuItem)
```

Types:

**Option** = {text, string() } | {help, string() } | {kind, WxItemKind}

**WxItemKind** = integer()

See [external documentation](#).

WxItemKind is one of ?wxITEM\_SEPARATOR | ?wxITEM\_NORMAL | ?wxITEM\_CHECK | ?wxITEM\_RADIO  
| ?wxITEM\_MAX

```
prepend(This::wxMenu(), Itemid::integer(), Text::string(), Submenu::wxMenu())
-> wxMenuItem() (see module wxMenuItem)
```

Equivalent to *prepend(This, Itemid, Text, Submenu, [])*.

```
prepend(This::wxMenu(), Itemid::integer(), Text::string(), X::string() |
wxMenu(), X::bool() | term()) -> ok | wxMenuItem() (see module wxMenuItem)
```

See [external documentation](#).

Alternatives:

```
prepend(This::wxMenu(), Itemid::integer(), Text::string(), Help::string(),
IsCheckable::bool()) -> ok
```

```
prepend(This::wxMenu(), Itemid::integer(), Text::string(), Submenu::wxMenu(),
[Option]) -> wxMenuItem:wxMenuItem()
```

Option = {help, string() }

```
prependCheckItem(This::wxMenu(), Itemid::integer(), Text::string()) ->
wxMenuItem() (see module wxMenuItem)
```

Equivalent to *prependCheckItem(This, Itemid, Text, [])*.

```
prependCheckItem(This::wxMenu(), Itemid::integer(), Text::string(), Options::
[Option]) -> wxMenuItem() (see module wxMenuItem)
```

Types:

**Option** = {help, string() }

See [external documentation](#).

```
prependRadioItem(This::wxMenu(), Itemid::integer(), Text::string()) ->
wxMenuItem() (see module wxMenuItem)
```

Equivalent to *prependRadioItem(This, Itemid, Text, [])*.

```
prependRadioItem(This::wxMenu(), Itemid::integer(), Text::string(), Options::
[Option]) -> wxMenuItem() (see module wxMenuItem)
```

Types:

**Option** = {help, string() }

See [external documentation](#).

```
prependSeparator(This::wxMenu()) -> wxMenuItem() (see module wxMenuItem)
```

See [external documentation](#).

```
remove(This::wxMenu(), X::integer() | term()) -> wxMenuItem() (see module
wxMenuItem)
```

See [external documentation](#).

Alternatives:

```
remove(This::wxMenu(), Itemid::integer()) -> wxMenuItem:wxMenuItem()
```

```
remove(This::wxMenu(), Item:wxMenuItem:wxMenuItem()) ->
wxMenuItem:wxMenuItem()
```

```
setHelpString(This::wxMenu(), Itemid::integer(), HelpString::string()) -> ok
```

See [external documentation](#).

```
setLabel(This::wxMenu(), Itemid::integer(), Label::string()) -> ok
```

See [external documentation](#).

## **wxMenu**

---

**setTitle(This::wxMenu(), Title::string()) -> ok**

See **external documentation**.

**destroy(This::wxMenu()) -> ok**

Destroys this object, do not use object again

# wxMenuBar

Erlang module

See external documentation: **wxMenuBar**.

This class is derived (and can use functions) from:

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxMenuBar()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxMenuBar()`

See external documentation.

`new(Style::integer()) -> wxMenuBar()`

See external documentation.

`append(This::wxMenuBar(), Menu::wxMenu() (see module wxMenu), Title::string()) -> bool()`

See external documentation.

`check(This::wxMenuBar(), Itemid::integer(), Check::bool()) -> ok`

See external documentation.

`enable(This::wxMenuBar()) -> bool()`

Equivalent to `enable(This, [])`.

`enable(This::wxMenuBar(), Options::[Option]) -> bool()`

Types:

**Option** = {enable, bool()}

See external documentation.

`enable(This::wxMenuBar(), Itemid::integer(), Enable::bool()) -> ok`

See external documentation.

`enableTop(This::wxMenuBar(), Pos::integer(), Flag::bool()) -> ok`

See external documentation.

`findMenu(This::wxMenuBar(), Title::string()) -> integer()`

See external documentation.

`findMenuItem(This::wxMenuBar(), MenuString::string(), ItemString::string()) -> integer()`

See external documentation.

`findItem(This::wxMenuBar(), Id::integer()) -> wxMenuItem()` (see module `wxMenuItem`)

See external documentation.

`getHelpString(This::wxMenuBar(), Itemid::integer()) -> string()`

See external documentation.

`getLabel(This::wxMenuBar()) -> string()`

See external documentation.

`getLabel(This::wxMenuBar(), Itemid::integer()) -> string()`

See external documentation.

`getLabelTop(This::wxMenuBar(), Pos::integer()) -> string()`

See external documentation.

`getMenu(This::wxMenuBar(), Pos::integer()) -> wxMenu()` (see module `wxMenu`)

See external documentation.

`getMenuCount(This::wxMenuBar()) -> integer()`

See external documentation.

`insert(This::wxMenuBar(), Pos::integer(), Menu::wxMenu())` (see module `wxMenu`), `Title::string()` -> `bool()`

See external documentation.

`isChecked(This::wxMenuBar(), Itemid::integer()) -> bool()`

See external documentation.

`isEnabled(This::wxMenuBar()) -> bool()`

See external documentation.

`isEnabled(This::wxMenuBar(), Itemid::integer()) -> bool()`

See external documentation.



`remove(This::wxMenuBar(), Pos::integer()) -> wxMenu()` (see module `wxMenu`)

See external documentation.

`replace(This::wxMenuBar(), Pos::integer(), Menu::wxMenu()` (see module `wxMenu`), `Title::string()`) -> `wxMenu()` (see module `wxMenu`)

See external documentation.

`setHelpString(This::wxMenuBar(), Itemid::integer(), HelpString::string()) -> ok`

See external documentation.

`setLabel(This::wxMenuBar(), S::string()) -> ok`

See external documentation.

`setLabel(This::wxMenuBar(), Itemid::integer(), Label::string()) -> ok`

See external documentation.

`setLabelTop(This::wxMenuBar(), Pos::integer(), Label::string()) -> ok`

See external documentation.

`destroy(This::wxMenuBar()) -> ok`

Destroys this object, do not use object again

## wxMenuEvent

---

Erlang module

See external documentation: **wxMenuEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*menu\_open, menu\_close, menu\_highlight*

See also the message variant *#wxMenu{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

`wxMenuEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getMenu(This::wxMenuEvent()) -> wxMenu()` (see module `wxMenu`)

See external documentation.

`getMenuId(This::wxMenuEvent()) -> integer()`

See external documentation.

`isPopup(This::wxMenuEvent()) -> bool()`

See external documentation.

## wxMenuItem

---

Erlang module

See external documentation: **wxMenuItem**.

### DATA TYPES

`wxMenuItem()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxMenuItem()`**

Equivalent to `new([])`.

**`new(Options::[Option])`** -> **`wxMenuItem()`**

Types:

**Option** = {parentMenu, wxMenu() (see module wxMenu)} | {id, integer()} | {text, string()} | {help, string()} | {kind, WxItemKind} | {subMenu, wxMenu() (see module wxMenu)}

**WxItemKind** = integer()

See external documentation.

WxItemKind is one of ?wxITEM\_SEPARATOR | ?wxITEM\_NORMAL | ?wxITEM\_CHECK | ?wxITEM\_RADIO | ?wxITEM\_MAX

**`check(This::wxMenuItem())`** -> **`ok`**

Equivalent to `check(This, [])`.

**`check(This::wxMenuItem(), Options::[Option])`** -> **`ok`**

Types:

**Option** = {check, bool()}

See external documentation.

**`enable(This::wxMenuItem())`** -> **`ok`**

Equivalent to `enable(This, [])`.

**`enable(This::wxMenuItem(), Options::[Option])`** -> **`ok`**

Types:

**Option** = {enable, bool()}

See external documentation.

**`getBitmap(This::wxMenuItem())`** -> **`wxBitmap()`** (see module wxBitmap)

See external documentation.

`getHelp(This::wxMenuItem()) -> string()`

See external documentation.

`getId(This::wxMenuItem()) -> integer()`

See external documentation.

`getKind(This::wxMenuItem()) -> WxItemKind`

Types:

`WxItemKind = integer()`

See external documentation.

WxItemKind is one of ?wxITEM\_SEPARATOR | ?wxITEM\_NORMAL | ?wxITEM\_CHECK | ?wxITEM\_RADIO | ?wxITEM\_MAX

`getLabel(This::wxMenuItem()) -> string()`

See external documentation.

`getLabelFromText(Text::string()) -> string()`

See external documentation.

`getMenu(This::wxMenuItem()) -> wxMenu() (see module wxMenu)`

See external documentation.

`getText(This::wxMenuItem()) -> string()`

See external documentation.

`getSubMenu(This::wxMenuItem()) -> wxMenu() (see module wxMenu)`

See external documentation.

`isCheckable(This::wxMenuItem()) -> bool()`

See external documentation.

`isChecked(This::wxMenuItem()) -> bool()`

See external documentation.

`isEnabled(This::wxMenuItem()) -> bool()`

See external documentation.

`isSeparator(This::wxMenuItem()) -> bool()`

See external documentation.

`isSubMenu(This::wxMenuItem()) -> bool()`

See external documentation.

**setBitmap**(This::wxMenuItem(), Bitmap:wxBitmap() (see module wxBitmap)) -> ok

See external documentation.

**setHelp**(This::wxMenuItem(), Str:string()) -> ok

See external documentation.

**setMenu**(This::wxMenuItem(), Menu:wxMenu() (see module wxMenu)) -> ok

See external documentation.

**setSubMenu**(This::wxMenuItem(), Menu:wxMenu() (see module wxMenu)) -> ok

See external documentation.

**setText**(This::wxMenuItem(), Str:string()) -> ok

See external documentation.

**destroy**(This::wxMenuItem()) -> ok

Destroys this object, do not use object again

## wxMessageDialog

---

Erlang module

See external documentation: **wxMessageDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxMessageDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Parent::wxWindow() (see module wxWindow), Message::string()) -> wxMessageDialog()`

Equivalent to `new(Parent, Message, [])`.

`new(Parent::wxWindow() (see module wxWindow), Message::string(), Options::[Option]) -> wxMessageDialog()`

Types:

**Option** = {caption, string()} | {style, integer()} | {pos, {X::integer(), Y::integer()}}

See **external documentation**.

`destroy(This::wxMessageDialog()) -> ok`

Destroys this object, do not use object again

# wxMiniFrame

Erlang module

See external documentation: **wxMiniFrame**.

This class is derived (and can use functions) from:

*wxFrame*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxMiniFrame()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxMiniFrame()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string()) -> wxMiniFrame()`

Equivalent to `new(Parent, Id, Title, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string(), Options::[Option]) -> wxMiniFrame()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxMiniFrame(), Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string()) -> bool()`

Equivalent to `create(This, Parent, Id, Title, [])`.

`create(This::wxMiniFrame(), Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`destroy(This::wxMiniFrame()) -> ok`

Destroys this object, do not use object again

## wxMirrorDC

---

Erlang module

See external documentation: **wxMirrorDC**.

This class is derived (and can use functions) from:  
*wxDC*

### DATA TYPES

`wxMirrorDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Dc::wxDC() (see module wxDC), Mirror::bool()) -> wxMirrorDC()`

See **external documentation**.

`destroy(This::wxMirrorDC()) -> ok`

Destroys this object, do not use object again



## wxMouseCaptureChangedEvent

---

Erlang module

See external documentation: **wxMouseCaptureChangedEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*mouse\_capture\_changed*

See also the message variant *#wxMouseCaptureChanged{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxMouseCaptureChangedEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`getCapturedWindow(This::wxMouseCaptureChangedEvent()) -> wxWindow()` (see module `wxWindow`)

See external documentation.

## wxMouseEvent

---

Erlang module

See external documentation: **wxMouseEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*left\_down, left\_up, middle\_down, middle\_up, right\_down, right\_up, motion, enter\_window, leave\_window, left\_dclick, middle\_dclick, right\_dclick, mousewheel, nc\_left\_down, nc\_left\_up, nc\_middle\_down, nc\_middle\_up, nc\_right\_down, nc\_right\_up, nc\_motion, nc\_enter\_window, nc\_leave\_window, nc\_left\_dclick, nc\_middle\_dclick, nc\_right\_dclick*

See also the message variant *#wxMouse{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

*wxMouseEvent()*

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

**altDown(This::wxMouseEvent()) -> bool()**

See external documentation.

**button(This::wxMouseEvent(), But::integer()) -> bool()**

See external documentation.

**buttonDClick(This::wxMouseEvent()) -> bool()**

Equivalent to *buttonDClick(This, [])*.

**buttonDClick(This::wxMouseEvent(), Options::[Option]) -> bool()**

Types:

**Option = {but, integer()}**

See external documentation.

**buttonDown(This::wxMouseEvent()) -> bool()**

Equivalent to *buttonDown(This, [])*.

**buttonDown(This::wxMouseEvent(), Options::[Option]) -> bool()**

Types:

**Option = {but, integer()}**

See external documentation.

`buttonUp(This::wxMouseEvent()) -> bool()`

Equivalent to `buttonUp(This, [])`.

`buttonUp(This::wxMouseEvent(), Options::[Option]) -> bool()`

Types:

`Option = {but, integer()}`

See external documentation.

`cmdDown(This::wxMouseEvent()) -> bool()`

See external documentation.

`controlDown(This::wxMouseEvent()) -> bool()`

See external documentation.

`dragging(This::wxMouseEvent()) -> bool()`

See external documentation.

`entering(This::wxMouseEvent()) -> bool()`

See external documentation.

`getButton(This::wxMouseEvent()) -> integer()`

See external documentation.

`getPosition(This::wxMouseEvent()) -> {X::integer(), Y::integer()}`

See external documentation.

`getLogicalPosition(This::wxMouseEvent(), Dc::wxDC() (see module wxDC)) -> {X::integer(), Y::integer()}`

See external documentation.

`getLinesPerAction(This::wxMouseEvent()) -> integer()`

See external documentation.

`getWheelRotation(This::wxMouseEvent()) -> integer()`

See external documentation.

`getWheelDelta(This::wxMouseEvent()) -> integer()`

See external documentation.

`getX(This::wxMouseEvent()) -> integer()`

See external documentation.

`getY(This::wxMouseEvent()) -> integer()`

See external documentation.

`isButton(This::wxMouseEvent()) -> bool()`

See external documentation.

`isPageScroll(This::wxMouseEvent()) -> bool()`

See external documentation.

`leaving(This::wxMouseEvent()) -> bool()`

See external documentation.

`leftDClick(This::wxMouseEvent()) -> bool()`

See external documentation.

`leftDown(This::wxMouseEvent()) -> bool()`

See external documentation.

`leftIsDown(This::wxMouseEvent()) -> bool()`

See external documentation.

`leftUp(This::wxMouseEvent()) -> bool()`

See external documentation.

`metaDown(This::wxMouseEvent()) -> bool()`

See external documentation.

`middleDClick(This::wxMouseEvent()) -> bool()`

See external documentation.

`middleDown(This::wxMouseEvent()) -> bool()`

See external documentation.

`middleIsDown(This::wxMouseEvent()) -> bool()`

See external documentation.

`middleUp(This::wxMouseEvent()) -> bool()`

See external documentation.

`moving(This::wxMouseEvent()) -> bool()`

See external documentation.

**rightDClick**(This::wxMouseEvent()) -> bool()

See external documentation.

**rightDown**(This::wxMouseEvent()) -> bool()

See external documentation.

**rightIsDown**(This::wxMouseEvent()) -> bool()

See external documentation.

**rightUp**(This::wxMouseEvent()) -> bool()

See external documentation.

**shiftDown**(This::wxMouseEvent()) -> bool()

See external documentation.

## wxMoveEvent

---

Erlang module

See external documentation: **wxMoveEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*move*

See also the message variant *#wxMove{}* event record type.

This class is derived (and can use functions) from:

*wxEvt*

## DATA TYPES

`wxMoveEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getPosition(This::wxMoveEvent()) -> {X::integer(), Y::integer()}`

See **external documentation**.

## wxMultiChoiceDialog

---

Erlang module

See external documentation: **wxMultiChoiceDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxMultiChoiceDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxMultiChoiceDialog()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Message::string(),  
Caption::string(), Choices::[[string()]]) -> wxMultiChoiceDialog()`

Equivalent to `new(Parent, Message, Caption, Choices, [])`.

`new(Parent::wxWindow() (see module wxWindow), Message::string(),  
Caption::string(), Choices::[[string()]], Options::[Option]) ->  
wxMultiChoiceDialog()`

Types:

**Option** = {style, integer()} | {pos, {X::integer(), Y::integer()}}

See external documentation.

`getSelections(This::wxMultiChoiceDialog()) -> [integer()]`

See external documentation.

`setSelections(This::wxMultiChoiceDialog(), Selections::[integer()]) -> ok`

See external documentation.

`destroy(This::wxMultiChoiceDialog()) -> ok`

Destroys this object, do not use object again

## wxNavigationKeyEvent

---

Erlang module

See external documentation: **wxNavigationKeyEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*navigation\_key*

See also the message variant *#wxNavigationKey{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxNavigationKeyEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

### Exports

`getDirection(This::wxNavigationKeyEvent()) -> bool()`

See external documentation.

`setDirection(This::wxNavigationKeyEvent(), BForward::bool()) -> ok`

See external documentation.

`isWindowChange(This::wxNavigationKeyEvent()) -> bool()`

See external documentation.

`setWindowChange(This::wxNavigationKeyEvent(), BIs::bool()) -> ok`

See external documentation.

`isFromTab(This::wxNavigationKeyEvent()) -> bool()`

See external documentation.

`setFromTab(This::wxNavigationKeyEvent(), BIs::bool()) -> ok`

See external documentation.

`getCurrentFocus(This::wxNavigationKeyEvent()) -> wxWindow() (see module wxWindow)`

See external documentation.

`setCurrentFocus(This::wxNavigationKeyEvent(), Win::wxWindow() (see module wxWindow)) -> ok`

See external documentation.



## wxNcPaintEvent

---

Erlang module

See external documentation: **wxNcPaintEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*nc\_paint*

See also the message variant *#wxNcPaint{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxNcPaintEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxNotebook

---

Erlang module

See external documentation: **wxNotebook**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxNotebook()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxNotebook()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Winid::integer()) -> wxNotebook()`

Equivalent to `new(Parent, Winid, [])`.

`new(Parent::wxWindow() (see module wxWindow), Winid::integer(), Options::[Option]) -> wxNotebook()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`addPage(This::wxNotebook(), Page::wxWindow() (see module wxWindow), Text::string()) -> bool()`

Equivalent to `addPage(This, Page, Text, [])`.

`addPage(This::wxNotebook(), Page::wxWindow() (see module wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

**Option** = {bSelect, bool()} | {imageId, integer()}

See external documentation.

`advanceSelection(This::wxNotebook()) -> ok`

Equivalent to `advanceSelection(This, [])`.

`advanceSelection(This::wxNotebook(), Options::[Option]) -> ok`

Types:

`Option = {forward, bool()}`

See external documentation.

`assignImageList(This::wxNotebook(), ImageList::wxImageList() (see module wxImageList)) -> ok`

See external documentation.

`create(This::wxNotebook(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, [])`.

`create(This::wxNotebook(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()`

Types:

`Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}`

See external documentation.

`deleteAllPages(This::wxNotebook()) -> bool()`

See external documentation.

`deletePage(This::wxNotebook(), NPage::integer()) -> bool()`

See external documentation.

`removePage(This::wxNotebook(), NPage::integer()) -> bool()`

See external documentation.

`getCurrentPage(This::wxNotebook()) -> wxWindow() (see module wxWindow)`

See external documentation.

`getImageList(This::wxNotebook()) -> wxImageList() (see module wxImageList)`

See external documentation.

`getPage(This::wxNotebook(), N::integer()) -> wxWindow() (see module wxWindow)`

See external documentation.

`getPageCount(This::wxNotebook()) -> integer()`

See external documentation.

`getPageImage(This::wxNotebook(), NPage::integer()) -> integer()`

See external documentation.

`getPageText(This::wxNotebook(), NPage::integer()) -> string()`

See external documentation.

`getRowCount(This::wxNotebook()) -> integer()`

See external documentation.

`getSelection(This::wxNotebook()) -> integer()`

See external documentation.

`getThemeBackgroundColour(This::wxNotebook()) -> colour() (see module wx)`

See external documentation.

`hitTest(This::wxNotebook(), Pt::{X::integer(), Y::integer()}) -> {integer(),  
Flags::integer()}`

See external documentation.

`insertPage(This::wxNotebook(), Position::integer(), Win::wxWindow() (see  
module wxWindow), StrText::string()) -> bool()`

Equivalent to `insertPage(This, Position, Win, StrText, [])`.

`insertPage(This::wxNotebook(), Position::integer(), Win::wxWindow() (see  
module wxWindow), StrText::string(), Options::[Option]) -> bool()`

Types:

`Option = {bSelect, bool()} | {imageId, integer()}`

See external documentation.

`setImageList(This::wxNotebook(), ImageList::wxImageList() (see module  
wxImageList)) -> ok`

See external documentation.

`setPadding(This::wxNotebook(), Padding::{W::integer(), H::integer()}) -> ok`

See external documentation.

`setPageSize(This::wxNotebook(), Size::{W::integer(), H::integer()}) -> ok`

See external documentation.

`setPageImage(This::wxNotebook(), NPage::integer(), NImage::integer()) ->  
bool()`

See external documentation.

`setPageText(This::wxNotebook(), NPage::integer(), StrText::string()) ->  
bool()`

See external documentation.

**setSelection(This::wxNotebook(), NPage::integer()) -> integer()**

See **external documentation**.

**changeSelection(This::wxNotebook(), NPage::integer()) -> integer()**

See **external documentation**.

**destroy(This::wxNotebook()) -> ok**

Destroys this object, do not use object again

## wxNotebookEvent

---

Erlang module

See external documentation: **wxNotebookEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*command\_notebook\_page\_changed, command\_notebook\_page\_changing*

See also the message variant *#wxNotebook{}* event record type.

This class is derived (and can use functions) from:

*wxNotifyEvent*

*wxCommandEvent*

*wxEvent*

## DATA TYPES

`wxNotebookEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getOldSelection(This::wxNotebookEvent()) -> integer()`

See external documentation.

`getSelection(This::wxNotebookEvent()) -> integer()`

See external documentation.

`setOldSelection(This::wxNotebookEvent(), NOldSel::integer()) -> ok`

See external documentation.

`setSelection(This::wxNotebookEvent(), NSel::integer()) -> ok`

See external documentation.

## wxNotifyEvent

---

Erlang module

See external documentation: **wxNotifyEvent**.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

### DATA TYPES

`wxNotifyEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`allow(This::wxNotifyEvent()) -> ok`

See external documentation.

`isAllowed(This::wxNotifyEvent()) -> bool()`

See external documentation.

`veto(This::wxNotifyEvent()) -> ok`

See external documentation.

## wxPageSetupDialog

---

Erlang module

See external documentation: **wxPageSetupDialog**.

### DATA TYPES

`wxPageSetupDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Parent::wxWindow() (see module wxWindow)) -> wxPageSetupDialog()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxPageSetupDialog()`

Types:

**Option** = {data, wxPageSetupDialogData() (see module wxPageSetupDialogData)}

See external documentation.

`getPageSetupData(This::wxPageSetupDialog()) -> wxPageSetupDialogData() (see module wxPageSetupDialogData)`

See external documentation.

`showModal(This::wxPageSetupDialog()) -> integer()`

See external documentation.

`destroy(This::wxPageSetupDialog()) -> ok`

Destroys this object, do not use object again



# wxPageSetupDialogData

---

Erlang module

See external documentation: **wxPageSetupDialogData**.

## DATA TYPES

`wxPageSetupDialogData()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxPageSetupDialogData()`

See external documentation.

`new(PrintData::wxPrintData() (see module wxPrintData) |  
wxPageSetupDialogData()) -> wxPageSetupDialogData()`

See external documentation.

`enableHelp(This::wxPageSetupDialogData(), Flag::bool()) -> ok`

See external documentation.

`enableMargins(This::wxPageSetupDialogData(), Flag::bool()) -> ok`

See external documentation.

`enableOrientation(This::wxPageSetupDialogData(), Flag::bool()) -> ok`

See external documentation.

`enablePaper(This::wxPageSetupDialogData(), Flag::bool()) -> ok`

See external documentation.

`enablePrinter(This::wxPageSetupDialogData(), Flag::bool()) -> ok`

See external documentation.

`getDefaultMinMargins(This::wxPageSetupDialogData()) -> bool()`

See external documentation.

`getEnableMargins(This::wxPageSetupDialogData()) -> bool()`

See external documentation.

`getEnableOrientation(This::wxPageSetupDialogData()) -> bool()`

See external documentation.

`getEnablePaper(This::wxPageSetupDialogData()) -> bool()`

See external documentation.

`getEnablePrinter(This::wxPageSetupDialogData()) -> bool()`

See external documentation.

`getEnableHelp(This::wxPageSetupDialogData()) -> bool()`

See external documentation.

`getDefaultInfo(This::wxPageSetupDialogData()) -> bool()`

See external documentation.

`getMarginTopLeft(This::wxPageSetupDialogData()) -> {X::integer(),  
Y::integer() }`

See external documentation.

`getMarginBottomRight(This::wxPageSetupDialogData()) -> {X::integer(),  
Y::integer() }`

See external documentation.

`getMinMarginTopLeft(This::wxPageSetupDialogData()) -> {X::integer(),  
Y::integer() }`

See external documentation.

`getMinMarginBottomRight(This::wxPageSetupDialogData()) -> {X::integer(),  
Y::integer() }`

See external documentation.

`getPaperId(This::wxPageSetupDialogData()) -> integer()`

See external documentation.

`getPaperSize(This::wxPageSetupDialogData()) -> {W::integer(), H::integer() }`

See external documentation.

`getPrintData(This::wxPageSetupDialogData()) -> wxPrintData() (see module  
wxPrintData)`

See external documentation.

`isOk(This::wxPageSetupDialogData()) -> bool()`

See external documentation.

`setDefaultInfo(This::wxPageSetupDialogData(), Flag::bool()) -> ok`

See external documentation.

**setDefaultMinMargins**(This::wxPageSetupDialogData(), Flag::bool()) -> ok

See external documentation.

**setMarginTopLeft**(This::wxPageSetupDialogData(), Pt::{X::integer(), Y::integer()}) -> ok

See external documentation.

**setMarginBottomRight**(This::wxPageSetupDialogData(), Pt::{X::integer(), Y::integer()}) -> ok

See external documentation.

**setMinMarginTopLeft**(This::wxPageSetupDialogData(), Pt::{X::integer(), Y::integer()}) -> ok

See external documentation.

**setMinMarginBottomRight**(This::wxPageSetupDialogData(), Pt::{X::integer(), Y::integer()}) -> ok

See external documentation.

**setPaperId**(This::wxPageSetupDialogData(), Id::integer()) -> ok

See external documentation.

**setPaperSize**(This::wxPageSetupDialogData(), X::integer() | term()) -> ok

See external documentation.

Alternatives:

**setPaperSize**(This::wxPageSetupDialogData(), Id::integer()) -> ok

**setPaperSize**(This::wxPageSetupDialogData(), Sz::{W::integer(), H::integer()}) -> ok

**setPrintData**(This::wxPageSetupDialogData(), PrintData::wxPrintData() (see module wxPrintData)) -> ok

See external documentation.

**destroy**(This::wxPageSetupDialogData()) -> ok

Destroys this object, do not use object again

## wxPaintDC

---

Erlang module

See external documentation: **wxPaintDC**.

This class is derived (and can use functions) from:

*wxWindowDC*

*wxDC*

## DATA TYPES

`wxPaintDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

**`new()`** -> **`wxPaintDC()`**

See **external documentation**.

**`new(Win::wxWindow())`** (see module **`wxWindow`**) -> **`wxPaintDC()`**

See **external documentation**.

**`destroy(This::wxPaintDC())`** -> **`ok`**

Destroys this object, do not use object again

## wxPaintEvent

---

Erlang module

See external documentation: **wxPaintEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*paint, paint\_icon*

See also the message variant *#wxPaint{}* event record type.

This class is derived (and can use functions) from:  
*wxEvent*

### DATA TYPES

`wxPaintEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## wxPalette

---

Erlang module

See external documentation: **wxPalette**.

### DATA TYPES

`wxPalette()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxPalette()`

See external documentation.

`new(Red::binary(), Green::binary(), Blue::binary()) -> wxPalette()`

See external documentation.

`create(This::wxPalette(), Red::binary(), Green::binary(), Blue::binary()) -> bool()`

See external documentation.

`getColoursCount(This::wxPalette()) -> integer()`

See external documentation.

`getPixel(This::wxPalette(), Red::integer(), Green::integer(), Blue::integer()) -> integer()`

See external documentation.

`getRGB(This::wxPalette(), Pixel::integer()) -> {bool(), Red::integer(), Green::integer(), Blue::integer()}`

See external documentation.

`isOk(This::wxPalette()) -> bool()`

See external documentation.

`destroy(This::wxPalette()) -> ok`

Destroys this object, do not use object again

## wxPaletteChangedEvent

---

Erlang module

See external documentation: **wxPaletteChangedEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*palette\_changed*

See also the message variant *#wxPaletteChanged{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxPaletteChangedEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`setChangedWindow(This::wxPaletteChangedEvent(), Win::wxWindow()) (see module wxWindow) -> ok`

See external documentation.

`getChangedWindow(This::wxPaletteChangedEvent()) -> wxWindow() (see module wxWindow)`

See external documentation.

## wxPanel

---

Erlang module

See external documentation: **wxPanel**.

This class is derived (and can use functions) from:

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxPanel()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxPanel()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxPanel()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxPanel()`

Types:

**Option** = {winid, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), X::integer(), Y::integer(), Width::integer(), Height::integer()) -> wxPanel()`

Equivalent to `new(Parent, X, Y, Width, Height, [])`.

`new(Parent::wxWindow() (see module wxWindow), X::integer(), Y::integer(), Width::integer(), Height::integer(), Options::[Option]) -> wxPanel()`

Types:

**Option** = {style, integer()}

See external documentation.

`initDialog(This::wxPanel()) -> ok`

See external documentation.

`destroy(This::wxPanel()) -> ok`

Destroys this object, do not use object again



## wxPasswordEntryDialog

---

Erlang module

See external documentation: **wxPasswordEntryDialog**.

This class is derived (and can use functions) from:

*wxTextEntryDialog*

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxPasswordEntryDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Parent::wxWindow() (see module wxWindow), Message::string()) -> wxPasswordEntryDialog()`

Equivalent to `new(Parent, Message, [])`.

`new(Parent::wxWindow() (see module wxWindow), Message::string(), Options::[Option]) -> wxPasswordEntryDialog()`

Types:

**Option** = {caption, string()} | {value, string()} | {style, integer()} | {pos, {X::integer(), Y::integer()}}

See external documentation.

`destroy(This::wxPasswordEntryDialog()) -> ok`

Destroys this object, do not use object again

## wxPen

---

Erlang module

See external documentation: **wxPen**.

### DATA TYPES

`wxPen()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxPen()`

See external documentation.

`new(Colour::colour() (see module wx)) -> wxPen()`

Equivalent to `new(Colour, [])`.

`new(Colour::colour() (see module wx), Options::[Option]) -> wxPen()`

Types:

`Option = {width, integer()} | {style, integer()}`

See external documentation.

`getCap(This::wxPen()) -> integer()`

See external documentation.

`getColour(This::wxPen()) -> colour() (see module wx)`

See external documentation.

`getJoin(This::wxPen()) -> integer()`

See external documentation.

`getStyle(This::wxPen()) -> integer()`

See external documentation.

`getWidth(This::wxPen()) -> integer()`

See external documentation.

`isOk(This::wxPen()) -> bool()`

See external documentation.

**setCap(This::wxPen(), CapStyle::integer()) -> ok**

See external documentation.

**setColour(This::wxPen(), Colour::colour() (see module wx)) -> ok**

See external documentation.

**setColour(This::wxPen(), Red::integer(), Green::integer(), Blue::integer()) -> ok**

See external documentation.

**setJoin(This::wxPen(), JoinStyle::integer()) -> ok**

See external documentation.

**setStyle(This::wxPen(), Style::integer()) -> ok**

See external documentation.

**setWidth(This::wxPen(), Width::integer()) -> ok**

See external documentation.

**destroy(This::wxPen()) -> ok**

Destroys this object, do not use object again

## wxPickerBase

---

Erlang module

See external documentation: **wxPickerBase**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxPickerBase()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`setInternalMargin(This::wxPickerBase(), Newmargin::integer()) -> ok`

See external documentation.

`getInternalMargin(This::wxPickerBase()) -> integer()`

See external documentation.

`setTextCtrlProportion(This::wxPickerBase(), Prop::integer()) -> ok`

See external documentation.

`setPickerCtrlProportion(This::wxPickerBase(), Prop::integer()) -> ok`

See external documentation.

`getTextCtrlProportion(This::wxPickerBase()) -> integer()`

See external documentation.

`getPickerCtrlProportion(This::wxPickerBase()) -> integer()`

See external documentation.

`hasTextCtrl(This::wxPickerBase()) -> bool()`

See external documentation.

`getTextCtrl(This::wxPickerBase()) -> wxTextCtrl() (see module wxTextCtrl)`

See external documentation.

`isTextCtrlGrowable(This::wxPickerBase()) -> bool()`

See external documentation.

**setPickerCtrlGrowable(This::wxPickerBase()) -> ok**

Equivalent to *setPickerCtrlGrowable(This, [])*.

**setPickerCtrlGrowable(This::wxPickerBase(), Options::[Option]) -> ok**

Types:

**Option = {grow, bool()}**

See **external documentation**.

**setTextCtrlGrowable(This::wxPickerBase()) -> ok**

Equivalent to *setTextCtrlGrowable(This, [])*.

**setTextCtrlGrowable(This::wxPickerBase(), Options::[Option]) -> ok**

Types:

**Option = {grow, bool()}**

See **external documentation**.

**isPickerCtrlGrowable(This::wxPickerBase()) -> bool()**

See **external documentation**.

## wxPostScriptDC

---

Erlang module

See external documentation: **wxPostScriptDC**.

This class is derived (and can use functions) from:  
*wxDC*

### DATA TYPES

`wxPostScriptDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxPostScriptDC()`

See external documentation.

`new(PrintData::wxPrintData() (see module wxPrintData)) -> wxPostScriptDC()`

See external documentation.

`setResolution(Ppi::integer()) -> ok`

See external documentation.

`getResolution() -> integer()`

See external documentation.

`destroy(This::wxPostScriptDC()) -> ok`

Destroys this object, do not use object again

## wxPreviewCanvas

---

Erlang module

See external documentation: **wxPreviewCanvas**.

This class is derived (and can use functions) from:

*wxScrolledWindow*

*wxPanel*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxPreviewCanvas()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxPreviewControlBar

---

Erlang module

See external documentation: **wxPreviewControlBar**.

This class is derived (and can use functions) from:

*wxPanel*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxPreviewControlBar()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

```
new(Preview::wxPrintPreview() (see module wxPrintPreview),  
Buttons::integer(), Parent::wxWindow() (see module wxWindow)) ->  
wxPreviewControlBar()
```

Equivalent to *new(Preview, Buttons, Parent, [])*.

```
new(Preview::wxPrintPreview() (see module wxPrintPreview),  
Buttons::integer(), Parent::wxWindow() (see module wxWindow), Options::  
[Option]) -> wxPreviewControlBar()
```

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

```
createButtons(This::wxPreviewControlBar()) -> ok
```

See external documentation.

```
getPrintPreview(This::wxPreviewControlBar()) -> wxPrintPreview() (see module  
wxPrintPreview)
```

See external documentation.

```
getZoomControl(This::wxPreviewControlBar()) -> integer()
```

See external documentation.

```
setZoomControl(This::wxPreviewControlBar(), Zoom::integer()) -> ok
```

See external documentation.

```
destroy(This::wxPreviewControlBar()) -> ok
```

Destroys this object, do not use object again



## wxPreviewFrame

---

Erlang module

See external documentation: **wxPreviewFrame**.

This class is derived (and can use functions) from:

*wxFrame*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxPreviewFrame()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new(Preview::wxPrintPreview() (see module wxPrintPreview), Parent::wxWindow() (see module wxWindow)) -> wxPreviewFrame()`

Equivalent to `new(Preview, Parent, [])`.

`new(Preview::wxPrintPreview() (see module wxPrintPreview), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxPreviewFrame()`

Types:

**Option** = {title, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`createControlBar(This::wxPreviewFrame()) -> ok`

See external documentation.

`createCanvas(This::wxPreviewFrame()) -> ok`

See external documentation.

`initialize(This::wxPreviewFrame()) -> ok`

See external documentation.

`onCloseWindow(This::wxPreviewFrame(), Event::wxCloseEvent() (see module wxCloseEvent)) -> ok`

See external documentation.

`destroy(This::wxPreviewFrame()) -> ok`

Destroys this object, do not use object again

## wxPrintData

---

Erlang module

See external documentation: **wxPrintData**.

### DATA TYPES

`wxPrintData()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxPrintData()`

See external documentation.

`new(PrintData::wxPrintData()) -> wxPrintData()`

See external documentation.

`getCollate(This::wxPrintData()) -> bool()`

See external documentation.

`getBin(This::wxPrintData()) -> WxPrintBin`

Types:

**WxPrintBin = integer()**

See external documentation.

WxPrintBin is one of ?wxPRINTBIN\_DEFAULT | ?wxPRINTBIN\_ONLYONE | ?wxPRINTBIN\_LOWER | ?wxPRINTBIN\_MIDDLE | ?wxPRINTBIN\_MANUAL | ?wxPRINTBIN\_ENVELOPE | ?wxPRINTBIN\_ENVMANUAL | ?wxPRINTBIN\_AUTO | ?wxPRINTBIN\_TRACTOR | ?wxPRINTBIN\_SMALLFMT | ?wxPRINTBIN\_LARGEFORMAT | ?wxPRINTBIN\_LARGECAPACITY | ?wxPRINTBIN\_CASSETTE | ?wxPRINTBIN\_FORMSOURCE | ?wxPRINTBIN\_USER

`getColour(This::wxPrintData()) -> bool()`

See external documentation.

`getDuplex(This::wxPrintData()) -> WxDuplexMode`

Types:

**WxDuplexMode = integer()**

See external documentation.

WxDuplexMode is one of ?wxDUPLEX\_SIMPLEX | ?wxDUPLEX\_HORIZONTAL | ?wxDUPLEX\_VERTICAL

`getNoCopies(This::wxPrintData()) -> integer()`

See external documentation.

**getOrientation(This::wxPrintData()) -> integer()**

See external documentation.

**getPaperId(This::wxPrintData()) -> integer()**

See external documentation.

**getPrinterName(This::wxPrintData()) -> string()**

See external documentation.

**getQuality(This::wxPrintData()) -> integer()**

See external documentation.

**isOk(This::wxPrintData()) -> bool()**

See external documentation.

**setBin(This::wxPrintData(), Bin::WxPrintBin) -> ok**

Types:

**WxPrintBin = integer()**

See external documentation.

WxPrintBin is one of ?wxPRINTBIN\_DEFAULT | ?wxPRINTBIN\_ONLYONE | ?wxPRINTBIN\_LOWER | ?wxPRINTBIN\_MIDDLE | ?wxPRINTBIN\_MANUAL | ?wxPRINTBIN\_ENVELOPE | ?wxPRINTBIN\_ENVMANUAL | ?wxPRINTBIN\_AUTO | ?wxPRINTBIN\_TRACTOR | ?wxPRINTBIN\_SMALLFMT | ?wxPRINTBIN\_LARGEFORMAT | ?wxPRINTBIN\_LARGECAPACITY | ?wxPRINTBIN\_CASSETTE | ?wxPRINTBIN\_FORMSOURCE | ?wxPRINTBIN\_USER

**setCollate(This::wxPrintData(), Flag::bool()) -> ok**

See external documentation.

**setColour(This::wxPrintData(), Colour::bool()) -> ok**

See external documentation.

**setDuplex(This::wxPrintData(), Duplex::WxDuplexMode) -> ok**

Types:

**WxDuplexMode = integer()**

See external documentation.

WxDuplexMode is one of ?wxDUPLEX\_SIMPLEX | ?wxDUPLEX\_HORIZONTAL | ?wxDUPLEX\_VERTICAL

**setNoCopies(This::wxPrintData(), V::integer()) -> ok**

See external documentation.

**setOrientation(This::wxPrintData(), Orient::integer()) -> ok**

See external documentation.

## **wxPrintData**

---

**setPaperId(This::wxPrintData(), SizeId::integer()) -> ok**

See **external documentation**.

**setPrinterName(This::wxPrintData(), Name::string()) -> ok**

See **external documentation**.

**setQuality(This::wxPrintData(), Quality::integer()) -> ok**

See **external documentation**.

**destroy(This::wxPrintData()) -> ok**

Destroys this object, do not use object again

## wxPrintDialog

---

Erlang module

See external documentation: **wxPrintDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxPrintDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new(Parent::wxWindow() (see module wxWindow)) -> wxPrintDialog()`**

Equivalent to `new(Parent, [])`.

**`new(Parent::wxWindow() (see module wxWindow), X::term()) -> wxPrintDialog()`**

See external documentation.

Alternatives:

`new(Parent::wxWindow:wxWindow(), [Option]) -> wxPrintDialog()`

Option = {data, wxPrintDialogData:wxPrintDialogData()}

`new(Parent::wxWindow:wxWindow(), Data::wxPrintData:wxPrintData()) -> wxPrintDialog()`

**`getPrintDialogData(This::wxPrintDialog()) -> wxPrintDialogData() (see module wxPrintDialogData)`**

See external documentation.

**`getPrintDC(This::wxPrintDialog()) -> wxDC() (see module wxDC)`**

See external documentation.

**`destroy(This::wxPrintDialog()) -> ok`**

Destroys this object, do not use object again

## wxPrintDialogData

---

Erlang module

See external documentation: **wxPrintDialogData**.

### DATA TYPES

`wxPrintDialogData()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxPrintDialogData()`

See external documentation.

`new(DialogData::wxPrintDialogData() | wxPrintData() (see module wxPrintData))  
-> wxPrintDialogData()`

See external documentation.

`enableHelp(This::wxPrintDialogData(), Flag::bool()) -> ok`

See external documentation.

`enablePageNumbers(This::wxPrintDialogData(), Flag::bool()) -> ok`

See external documentation.

`enablePrintToFile(This::wxPrintDialogData(), Flag::bool()) -> ok`

See external documentation.

`enableSelection(This::wxPrintDialogData(), Flag::bool()) -> ok`

See external documentation.

`getAllPages(This::wxPrintDialogData()) -> bool()`

See external documentation.

`getCollate(This::wxPrintDialogData()) -> bool()`

See external documentation.

`getFromPage(This::wxPrintDialogData()) -> integer()`

See external documentation.

`getMaxPage(This::wxPrintDialogData()) -> integer()`

See external documentation.

`getMinPage(This::wxPrintDialogData()) -> integer()`

See external documentation.

`getNoCopies(This::wxPrintDialogData()) -> integer()`

See external documentation.

`getPrintData(This::wxPrintDialogData()) -> wxPrintData()` (see module `wxPrintData`)

See external documentation.

`getPrintToFile(This::wxPrintDialogData()) -> bool()`

See external documentation.

`getSelection(This::wxPrintDialogData()) -> bool()`

See external documentation.

`getToPage(This::wxPrintDialogData()) -> integer()`

See external documentation.

`isOk(This::wxPrintDialogData()) -> bool()`

See external documentation.

`setCollate(This::wxPrintDialogData(), Flag::bool()) -> ok`

See external documentation.

`setFromPage(This::wxPrintDialogData(), V::integer()) -> ok`

See external documentation.

`setMaxPage(This::wxPrintDialogData(), V::integer()) -> ok`

See external documentation.

`setMinPage(This::wxPrintDialogData(), V::integer()) -> ok`

See external documentation.

`setNoCopies(This::wxPrintDialogData(), V::integer()) -> ok`

See external documentation.

`setPrintData(This::wxPrintDialogData(), PrintData::wxPrintData()` (see module `wxPrintData`)) `-> ok`

See external documentation.

`setPrintToFile(This::wxPrintDialogData(), Flag::bool()) -> ok`

See external documentation.

## **wxPrintDialogData**

---

**setSelection(This::wxPrintDialogData(), Flag::bool()) -> ok**

See **external documentation**.

**setToPage(This::wxPrintDialogData(), V::integer()) -> ok**

See **external documentation**.

**destroy(This::wxPrintDialogData()) -> ok**

Destroys this object, do not use object again



# wxPrintPreview

---

Erlang module

See external documentation: **wxPrintPreview**.

## DATA TYPES

`wxPrintPreview()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new(Printout::wxPrintout() (see module wxPrintout)) -> wxPrintPreview()`

Equivalent to `new(Printout, [])`.

`new(Printout::wxPrintout() (see module wxPrintout), Options::[Option]) -> wxPrintPreview()`

Types:

`Option = {printoutForPrinting, wxPrintout() (see module wxPrintout)} | {data, wxPrintDialogData() (see module wxPrintDialogData)}`

See external documentation.

`new(Printout::wxPrintout() (see module wxPrintout),  
PrintoutForPrinting::wxPrintout() (see module wxPrintout),  
Data::wxPrintData() (see module wxPrintData)) -> wxPrintPreview()`

See external documentation.

`getCanvas(This::wxPrintPreview()) -> wxPreviewCanvas() (see module wxPreviewCanvas)`

See external documentation.

`getCurrentPage(This::wxPrintPreview()) -> integer()`

See external documentation.

`getFrame(This::wxPrintPreview()) -> wxFrame() (see module wxFrame)`

See external documentation.

`getMaxPage(This::wxPrintPreview()) -> integer()`

See external documentation.

`getMinPage(This::wxPrintPreview()) -> integer()`

See external documentation.

`getPrintout(This::wxPrintPreview()) -> wxPrintout()` (see module `wxPrintout`)

See external documentation.

`getPrintoutForPrinting(This::wxPrintPreview()) -> wxPrintout()` (see module `wxPrintout`)

See external documentation.

`isOk(This::wxPrintPreview()) -> bool()`

See external documentation.

`paintPage(This::wxPrintPreview(), Canvas::wxPreviewCanvas() (see module wxPreviewCanvas), Dc::wxDC() (see module wxDC)) -> bool()`

See external documentation.

`print(This::wxPrintPreview(), Interactive::bool()) -> bool()`

See external documentation.

`renderPage(This::wxPrintPreview(), PageNum::integer()) -> bool()`

See external documentation.

`setCanvas(This::wxPrintPreview(), Canvas::wxPreviewCanvas() (see module wxPreviewCanvas)) -> ok`

See external documentation.

`setCurrentPage(This::wxPrintPreview(), PageNum::integer()) -> bool()`

See external documentation.

`setFrame(This::wxPrintPreview(), Frame::wxFrame() (see module wxFrame)) -> ok`

See external documentation.

`setPrintout(This::wxPrintPreview(), Printout::wxPrintout() (see module wxPrintout)) -> ok`

See external documentation.

`setZoom(This::wxPrintPreview(), Percent::integer()) -> ok`

See external documentation.

`destroy(This::wxPrintPreview()) -> ok`

Destroys this object, do not use object again

## wxPrinter

---

Erlang module

See external documentation: **wxPrinter**.

### DATA TYPES

`wxPrinter()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxPrinter()`

Equivalent to `new([])`.

`new(Options::[Option]) -> wxPrinter()`

Types:

`Option = {data, wxPrintDialogData() (see module wxPrintDialogData)}`

See external documentation.

`createAbortWindow(This::wxPrinter(), Parent::wxWindow() (see module wxWindow), Printout::wxPrintout() (see module wxPrintout)) -> wxWindow() (see module wxWindow)`

See external documentation.

`getAbort(This::wxPrinter()) -> bool()`

See external documentation.

`getLastError() -> WxPrinterError`

Types:

`WxPrinterError = integer()`

See external documentation.

`WxPrinterError` is one of `?wxPRINTER_NO_ERROR` | `?wxPRINTER_CANCELLED` | `?wxPRINTER_ERROR`

`getPrintDialogData(This::wxPrinter()) -> wxPrintDialogData() (see module wxPrintDialogData)`

See external documentation.

`print(This::wxPrinter(), Parent::wxWindow() (see module wxWindow), Printout::wxPrintout() (see module wxPrintout)) -> bool()`

Equivalent to `print(This, Parent, Printout, [])`.

```
print(This::wxPrinter(), Parent::wxWindow() (see module wxWindow),  
Printout::wxPrintout() (see module wxPrintout), Options::[Option]) -> bool()
```

Types:

**Option** = {prompt, bool()}

See external documentation.

```
printDialog(This::wxPrinter(), Parent::wxWindow() (see module wxWindow)) ->  
wxDC() (see module wxDC)
```

See external documentation.

```
reportError(This::wxPrinter(), Parent::wxWindow() (see module wxWindow),  
Printout::wxPrintout() (see module wxPrintout), Message::string()) -> ok
```

See external documentation.

```
setup(This::wxPrinter(), Parent::wxWindow() (see module wxWindow)) -> bool()
```

See external documentation.

```
destroy(This::wxPrinter()) -> ok
```

Destroys this object, do not use object again

## wxPrintout

Erlang module

See external documentation: **wxPrintout**.

## DATA TYPES

`wxPrintout()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new(Title::string(), OnPrintPage::function()) -> wxPrintout()` (see module `wxPrintout`)

@equiv `new(Title, OnPrintPage, [])`

`new(Title::string(), OnPrintPage::function(), Opts::[Option]) -> wxPrintout()` (see module `wxPrintout`)

Types:

**Option** = {`onPreparePrinting`, `OnPreparePrinting::function()`} | {`onBeginPrinting`, `OnBeginPrinting::function()`} | {`onEndPrinting`, `OnEndPrinting::function()`} | {`onBeginDocument`, `OnBeginDocument::function()`} | {`onEndDocument`, `OnEndDocument::function()`} | {`hasPage`, `HasPage::function()`} | {`getPageInfo`, `GetPageInfo::function()`}

Creates a `wxPrintout` object with a callback fun and optionally other callback funs.

```
OnPrintPage(This,Page) -> boolean()
```

```
OnPreparePrinting(This) -> term()
```

```
OnBeginPrinting(This) -> term()
```

```
OnEndPrinting(This) -> term()
```

```
OnBeginDocument(This,StartPage,EndPage) -> boolean()
```

```
OnEndDocument(This) -> term()
```

```
HasPage(This,Page) -> boolean()
```

```
GetPageInfo(This) -> {MinPage::integer(), MaxPage::integer(), PageFrom::integer(), PageTo::integer() }
```

The *This* argument is the wxPrintout object reference to this object

NOTE: The callbacks may not call other processes.

`getDC(This::wxPrintout()) -> wxDC() (see module wxDC)`

See external documentation.

`getPageSizeMM(This::wxPrintout()) -> {W::integer(), H::integer()}`

See external documentation.

`getPageSizePixels(This::wxPrintout()) -> {W::integer(), H::integer()}`

See external documentation.

`getPaperRectPixels(This::wxPrintout()) -> {X::integer(), Y::integer(),  
W::integer(), H::integer()}`

See external documentation.

`getPPIPrinter(This::wxPrintout()) -> {X::integer(), Y::integer()}`

See external documentation.

`getPPIScreen(This::wxPrintout()) -> {X::integer(), Y::integer()}`

See external documentation.

`getTitle(This::wxPrintout()) -> string()`

See external documentation.

`isPreview(This::wxPrintout()) -> bool()`

See external documentation.

`fitThisSizeToPaper(This::wxPrintout(), ImageSize::{W::integer(),  
H::integer()}) -> ok`

See external documentation.

`fitThisSizeToPage(This::wxPrintout(), ImageSize::{W::integer(),  
H::integer()}) -> ok`

See external documentation.

`fitThisSizeToPageMargins(This::wxPrintout(), ImageSize::{W::integer(),  
H::integer()}, PageSetupData::wxPageSetupDialogData() (see module  
wxPageSetupDialogData)) -> ok`

See external documentation.

`mapScreenSizeToPaper(This::wxPrintout()) -> ok`

See external documentation.

**mapScreenSizeToPage(This::wxPrintout()) -> ok**

See external documentation.

**mapScreenSizeToPageMargins(This::wxPrintout(),  
PageSetupData::wxPageSetupDialogData() (see module wxPageSetupDialogData)) ->  
ok**

See external documentation.

**mapScreenSizeToDevice(This::wxPrintout()) -> ok**

See external documentation.

**getLogicalPaperRect(This::wxPrintout()) -> {X::integer(), Y::integer(),  
W::integer(), H::integer()}**

See external documentation.

**getLogicalPageRect(This::wxPrintout()) -> {X::integer(), Y::integer(),  
W::integer(), H::integer()}**

See external documentation.

**getLogicalPageMarginsRect(This::wxPrintout(),  
PageSetupData::wxPageSetupDialogData() (see module wxPageSetupDialogData)) ->  
{X::integer(), Y::integer(), W::integer(), H::integer()}**

See external documentation.

**setLogicalOrigin(This::wxPrintout(), X::integer(), Y::integer()) -> ok**

See external documentation.

**offsetLogicalOrigin(This::wxPrintout(), Xoff::integer(), Yoff::integer()) ->  
ok**

See external documentation.

**destroy(This::wxPrintout()) -> ok**

Destroys this object, do not use object again

## wxProgressDialog

---

Erlang module

See external documentation: **wxProgressDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxProgressDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Title::string(), Message::string()) -> wxProgressDialog()`

Equivalent to `new(Title, Message, [])`.

`new(Title::string(), Message::string(), Options::[Option]) -> wxProgressDialog()`

Types:

**Option** = {maximum, integer()} | {parent, wxWindow() (see module wxWindow)} | {style, integer()}

See external documentation.

`resume(This::wxProgressDialog()) -> ok`

See external documentation.

`update(This::wxProgressDialog()) -> ok`

See external documentation.

`update(This::wxProgressDialog(), Value::integer()) -> bool()`

Equivalent to `update(This, Value, [])`.

`update(This::wxProgressDialog(), Value::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {newmsg, string()}

See external documentation.

`destroy(This::wxProgressDialog()) -> ok`

Destroys this object, do not use object again



## wxQueryNewPaletteEvent

---

Erlang module

See external documentation: **wxQueryNewPaletteEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*query\_new\_palette*

See also the message variant *#wxQueryNewPalette{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxQueryNewPaletteEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`setPaletteRealized(This::wxQueryNewPaletteEvent(), Realized::bool()) -> ok`

See **external documentation**.

`getPaletteRealized(This::wxQueryNewPaletteEvent()) -> bool()`

See **external documentation**.

## wxRadioBox

---

Erlang module

See external documentation: **wxRadioBox**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxRadioBox()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

```
new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string(),
Pos::{X::integer(), Y::integer()}, Size::{W::integer(), H::integer()},
Choices::[[string()]]) -> wxRadioBox()
```

Equivalent to `new(Parent, Id, Title, Pos, Size, Choices, [])`.

```
new(Parent::wxWindow() (see module wxWindow), Id::integer(), Title::string(),
Pos::{X::integer(), Y::integer()}, Size::{W::integer(), H::integer()},
Choices::[[string()]], Options::[Option]) -> wxRadioBox()
```

Types:

**Option** = {majorDim, integer()} | {style, integer()} | {val, wx() (see module wx)}

See external documentation.

```
create(This::wxRadioBox(), Parent::wxWindow() (see module wxWindow),
Id::integer(), Title::string(), Pos::{X::integer(), Y::integer()}, Size::
{W::integer(), H::integer()}, Choices::[[string()]]) -> bool()
```

Equivalent to `create(This, Parent, Id, Title, Pos, Size, Choices, [])`.

```
create(This::wxRadioBox(), Parent::wxWindow() (see module wxWindow),
Id::integer(), Title::string(), Pos::{X::integer(), Y::integer()}, Size::
{W::integer(), H::integer()}, Choices::[[string()]], Options::[Option]) ->
bool()
```

Types:

**Option** = {majorDim, integer()} | {style, integer()} | {val, wx() (see module wx)}

See external documentation.

```
enable(This::wxRadioBox()) -> bool()
```

Equivalent to `enable(This, [])`.

**enable(This::wxRadioBox(), X::integer() | term()) -> bool()**

See external documentation.

Alternatives:

**enable(This::wxRadioBox(), N::integer()) -> enable(This,N, [])**

**enable(This::wxRadioBox(), [Option]) -> bool()**

Option = {enable, bool()}

**enable(This::wxRadioBox(), N::integer(), Options::[Option]) -> bool()**

Types:

**Option = {enable, bool()}**

See external documentation.

**getSelection(This::wxRadioBox()) -> integer()**

See external documentation.

**getString(This::wxRadioBox(), N::integer()) -> string()**

See external documentation.

**setSelection(This::wxRadioBox(), N::integer()) -> ok**

See external documentation.

**show(This::wxRadioBox()) -> bool()**

Equivalent to *show(This, [])*.

**show(This::wxRadioBox(), X::integer() | term()) -> bool()**

See external documentation.

Alternatives:

**show(This::wxRadioBox(), N::integer()) -> show(This,N, [])**

**show(This::wxRadioBox(), [Option]) -> bool()**

Option = {show, bool()}

**show(This::wxRadioBox(), N::integer(), Options::[Option]) -> bool()**

Types:

**Option = {show, bool()}**

See external documentation.

**getColumnCount(This::wxRadioBox()) -> integer()**

See external documentation.

**getItemHelpText(This::wxRadioBox(), N::integer()) -> string()**

See external documentation.

`getItemToolTip(This::wxRadioBox(), Item::integer()) -> wxToolTip()` (see module `wxToolTip`)

See external documentation.

`getItemFromPoint(This::wxRadioBox(), Pt::{X::integer(), Y::integer()}) -> integer()`

See external documentation.

`getRowCount(This::wxRadioBox()) -> integer()`

See external documentation.

`isItemEnabled(This::wxRadioBox(), N::integer()) -> bool()`

See external documentation.

`isItemShown(This::wxRadioBox(), N::integer()) -> bool()`

See external documentation.

`setItemHelpText(This::wxRadioBox(), N::integer(), HelpText::string()) -> ok`

See external documentation.

`setItemToolTip(This::wxRadioBox(), Item::integer(), Text::string()) -> ok`

See external documentation.

`destroy(This::wxRadioBox()) -> ok`

Destroys this object, do not use object again

## wxRadioButton

---

Erlang module

See external documentation: **wxRadioButton**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxRadioButton()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxRadioButton()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string())  
-> wxRadioButton()`

Equivalent to `new(Parent, Id, Label, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string(),  
Options::[Option]) -> wxRadioButton()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} |  
{validator, wx() (see module wx)}

See external documentation.

`create(This::wxRadioButton(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string()) -> bool()`

Equivalent to `create(This, Parent, Id, Label, [])`.

`create(This::wxRadioButton(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} |  
{validator, wx() (see module wx)}

See external documentation.

`getValue(This::wxRadioButton()) -> bool()`

See external documentation.

## **wxRadioButton**

---

**setValue(This::wxRadioButton(), Val::bool()) -> ok**

See **external documentation**.

**destroy(This::wxRadioButton()) -> ok**

Destroys this object, do not use object again

## wxRegion

---

Erlang module

See external documentation: **wxRegion**.

### DATA TYPES

`wxRegion()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()` -> `wxRegion()`**

See external documentation.

**`new(X::term())` -> `wxRegion()`**

See external documentation.

Alternatives:

`new(Bmp::wxBitmap:wxBitmap())` -> `wxRegion()`

`new(Rect::{X::integer(),Y::integer(),W::integer(),H::integer()})` -> `wxRegion()`

**`new(TopLeft::{X::integer(), Y::integer()}, BottomRight::{X::integer(), Y::integer()})` -> `wxRegion()`**

See external documentation.

**`new(X::integer(), Y::integer(), W::integer(), H::integer())` -> `wxRegion()`**

See external documentation.

**`clear(This::wxRegion())` -> `ok`**

See external documentation.

**`contains(This::wxRegion(), X::term())` -> `WxRegionContain`**

See external documentation.

Alternatives:

`contains(This::wxRegion(), Pt::{X::integer(),Y::integer()})` -> `WxRegionContain`  
`WxRegionContain = integer()`

`WxRegionContain` is one of `?wxOutRegion` | `?wxPartRegion` | `?wxInRegion`

`contains(This::wxRegion(), Rect::{X::integer(),Y::integer(),W::integer(),H::integer()})` -> `WxRegionContain`  
`WxRegionContain = integer()`

`WxRegionContain` is one of `?wxOutRegion` | `?wxPartRegion` | `?wxInRegion`

**contains(This::wxRegion(), X::integer(), Y::integer()) -> WxRegionContain**

Types:

**WxRegionContain = integer()**

See [external documentation](#).

WxRegionContain is one of ?wxOutRegion | ?wxPartRegion | ?wxInRegion

**contains(This::wxRegion(), X::integer(), Y::integer(), W::integer(),  
H::integer()) -> WxRegionContain**

Types:

**WxRegionContain = integer()**

See [external documentation](#).

WxRegionContain is one of ?wxOutRegion | ?wxPartRegion | ?wxInRegion

**convertToBitmap(This::wxRegion()) -> wxBitmap() (see module wxBitmap)**

See [external documentation](#).

**getBox(This::wxRegion()) -> {X::integer(), Y::integer(), W::integer(),  
H::integer() }**

See [external documentation](#).

**intersect(This::wxRegion(), X::wxRegion() | term()) -> bool()**

See [external documentation](#).

Alternatives:

**intersect(This::wxRegion(), Region::wxRegion()) -> bool()**

**intersect(This::wxRegion(), Rect::  
{X::integer(),Y::integer(),W::integer(),H::integer()}) -> bool()**

**intersect(This::wxRegion(), X::integer(), Y::integer(), W::integer(),  
H::integer()) -> bool()**

See [external documentation](#).

**isEmpty(This::wxRegion()) -> bool()**

See [external documentation](#).

**subtract(This::wxRegion(), X::wxRegion() | term()) -> bool()**

See [external documentation](#).

Alternatives:

**subtract(This::wxRegion(), Region::wxRegion()) -> bool()**

**subtract(This::wxRegion(), Rect::  
{X::integer(),Y::integer(),W::integer(),H::integer()}) -> bool()**

**subtract(This::wxRegion(), X::integer(), Y::integer(), W::integer(),  
H::integer()) -> bool()**

See [external documentation](#).



```
offset(This::wxRegion(), Pt::{X::integer(), Y::integer()}) -> bool()
```

See external documentation.

```
offset(This::wxRegion(), X::integer(), Y::integer()) -> bool()
```

See external documentation.

```
union(This::wxRegion(), X::term()) -> bool()
```

See external documentation.

Alternatives:

```
union(This::wxRegion(), Region::wxRegion() | wxBitmap:wxBitmap()) -> bool()
```

```
union(This::wxRegion(),                               Rect::
{X::integer(),Y::integer(),W::integer(),H::integer()}) -> bool()
```

```
union(This::wxRegion(), Bmp::wxBitmap() (see module wxBitmap),
Transp::colour() (see module wx)) -> bool()
```

Equivalent to *union(This, Bmp, Transp, [])*.

```
union(This::wxRegion(), Bmp::wxBitmap() (see module wxBitmap),
Transp::colour() (see module wx), Options::[Option]) -> bool()
```

Types:

**Option** = {tolerance, integer()}

See external documentation.

```
union(This::wxRegion(), X::integer(), Y::integer(), W::integer(),
H::integer()) -> bool()
```

See external documentation.

```
Xor(This::wxRegion(), X::wxRegion() | term()) -> bool()
```

See external documentation.

Alternatives:

```
'Xor'(This::wxRegion(), Region::wxRegion()) -> bool()
```

```
'Xor'(This::wxRegion(),                               Rect::
{X::integer(),Y::integer(),W::integer(),H::integer()}) -> bool()
```

```
Xor(This::wxRegion(), X::integer(), Y::integer(), W::integer(), H::integer())
-> bool()
```

See external documentation.

```
destroy(This::wxRegion()) -> ok
```

Destroys this object, do not use object again

## wxSashEvent

---

Erlang module

See external documentation: **wxSashEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*sash\_dragged*

See also the message variant *#wxSash{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

## DATA TYPES

`wxSashEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getEdge(This::wxSashEvent()) -> WxSashEdgePosition`

Types:

**WxSashEdgePosition** = integer()

See **external documentation**.

WxSashEdgePosition is one of ?wxSASH\_TOP | ?wxSASH\_RIGHT | ?wxSASH\_BOTTOM | ?wxSASH\_LEFT | ?wxSASH\_NONE

`getDragRect(This::wxSashEvent()) -> {X::integer(), Y::integer(), W::integer(), H::integer()}`

See **external documentation**.

`getDragStatus(This::wxSashEvent()) -> WxSashDragStatus`

Types:

**WxSashDragStatus** = integer()

See **external documentation**.

WxSashDragStatus is one of ?wxSASH\_STATUS\_OK | ?wxSASH\_STATUS\_OUT\_OF\_RANGE

## wxSashLayoutWindow

---

Erlang module

See external documentation: **wxSashLayoutWindow**.

This class is derived (and can use functions) from:

*wxSashWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxSashLayoutWindow()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxSashLayoutWindow()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxSashLayoutWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxSashLayoutWindow()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxSashLayoutWindow(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

`create(This::wxSashLayoutWindow(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`getAlignment(This::wxSashLayoutWindow()) -> WxLayoutAlignment`

Types:

**WxLayoutAlignment** = integer()

See **external documentation**.

WxLayoutAlignment is one of ?wxLAYOUT\_NONE | ?wxLAYOUT\_TOP | ?wxLAYOUT\_LEFT | ?wxLAYOUT\_RIGHT | ?wxLAYOUT\_BOTTOM

**getOrientation(This::wxSashLayoutWindow()) -> WxLayoutOrientation**

Types:

**WxLayoutOrientation = integer()**

See **external documentation**.

WxLayoutOrientation is one of ?wxLAYOUT\_HORIZONTAL | ?wxLAYOUT\_VERTICAL

**setAlignment(This::wxSashLayoutWindow(), Align::WxLayoutAlignment) -> ok**

Types:

**WxLayoutAlignment = integer()**

See **external documentation**.

WxLayoutAlignment is one of ?wxLAYOUT\_NONE | ?wxLAYOUT\_TOP | ?wxLAYOUT\_LEFT | ?wxLAYOUT\_RIGHT | ?wxLAYOUT\_BOTTOM

**setDefaultSize(This::wxSashLayoutWindow(), Size::{W::integer(), H::integer()}) -> ok**

See **external documentation**.

**setOrientation(This::wxSashLayoutWindow(), Orient::WxLayoutOrientation) -> ok**

Types:

**WxLayoutOrientation = integer()**

See **external documentation**.

WxLayoutOrientation is one of ?wxLAYOUT\_HORIZONTAL | ?wxLAYOUT\_VERTICAL

**destroy(This::wxSashLayoutWindow()) -> ok**

Destroys this object, do not use object again

# wxSashWindow

---

Erlang module

See external documentation: **wxSashWindow**.

This class is derived (and can use functions) from:

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxSashWindow()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxSashWindow()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxSashWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxSashWindow()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`getSashVisible(This::wxSashWindow(), Edge::WxSashEdgePosition) -> bool()`

Types:

**WxSashEdgePosition** = integer()

See external documentation.

WxSashEdgePosition is one of ?wxSASH\_TOP | ?wxSASH\_RIGHT | ?wxSASH\_BOTTOM | ?wxSASH\_LEFT | ?wxSASH\_NONE

`getMaximumSizeX(This::wxSashWindow()) -> integer()`

See external documentation.

`getMaximumSizeY(This::wxSashWindow()) -> integer()`

See external documentation.

`getMinimumSizeX(This::wxSashWindow()) -> integer()`

See external documentation.

**getMinimumSizeY(This::wxSashWindow()) -> integer()**

See [external documentation](#).

**setMaximumSizeX(This::wxSashWindow(), Max::integer()) -> ok**

See [external documentation](#).

**setMaximumSizeY(This::wxSashWindow(), Max::integer()) -> ok**

See [external documentation](#).

**setMinimumSizeX(This::wxSashWindow(), Min::integer()) -> ok**

See [external documentation](#).

**setMinimumSizeY(This::wxSashWindow(), Min::integer()) -> ok**

See [external documentation](#).

**setSashVisible(This::wxSashWindow(), Edge::WxSashEdgePosition, Sash::bool())  
-> ok**

Types:

**WxSashEdgePosition = integer()**

See [external documentation](#).

WxSashEdgePosition is one of ?wxSASH\_TOP | ?wxSASH\_RIGHT | ?wxSASH\_BOTTOM | ?wxSASH\_LEFT | ?wxSASH\_NONE

**destroy(This::wxSashWindow()) -> ok**

Destroys this object, do not use object again

## wxScreenDC

---

Erlang module

See external documentation: **wxScreenDC**.

This class is derived (and can use functions) from:  
*wxDC*

### DATA TYPES

`wxScreenDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxScreenDC()`**

See external documentation.

**`destroy(This::wxScreenDC())`** -> **`ok`**

Destroys this object, do not use object again

## wxScrollBar

---

Erlang module

See external documentation: **wxScrollBar**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxScrollBar()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxScrollBar()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxScrollBar()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxScrollBar()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxScrollBar(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, [])`.

`create(This::wxScrollBar(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`getRange(This::wxScrollBar()) -> integer()`

See external documentation.



`getPageSize(This::wxScrollBar()) -> integer()`

See external documentation.

`getThumbPosition(This::wxScrollBar()) -> integer()`

See external documentation.

`getThumbSize(This::wxScrollBar()) -> integer()`

See external documentation.

`setThumbPosition(This::wxScrollBar(), ViewStart::integer()) -> ok`

See external documentation.

`setScrollbar(This::wxScrollBar(), Position::integer(), ThumbSize::integer(), Range::integer(), PageSize::integer()) -> ok`

Equivalent to `setScrollbar(This, Position, ThumbSize, Range, PageSize, [])`.

`setScrollbar(This::wxScrollBar(), Position::integer(), ThumbSize::integer(), Range::integer(), PageSize::integer(), Options::[Option]) -> ok`

Types:

**Option** = {refresh, bool()}

See external documentation.

`destroy(This::wxScrollBar()) -> ok`

Destroys this object, do not use object again

## wxScrollEvent

---

Erlang module

See external documentation: **wxScrollEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*scroll\_top, scroll\_bottom, scroll\_lineup, scroll\_linedown, scroll\_pageup, scroll\_pagedown, scroll\_thumbtrack, scroll\_thumbrelease, scroll\_changed*

See also the message variant *#wxScroll{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

## DATA TYPES

`wxScrollEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

`getOrientation(This::wxScrollEvent()) -> integer()`

See **external documentation**.

`getPosition(This::wxScrollEvent()) -> integer()`

See **external documentation**.

## wxScrollWinEvent

---

Erlang module

See external documentation: **wxScrollWinEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*scrollwin\_top*, *scrollwin\_bottom*, *scrollwin\_lineup*, *scrollwin\_linedown*, *scrollwin\_pageup*,  
*scrollwin\_pagedown*, *scrollwin\_thumbtrack*, *scrollwin\_thumbrelease*

See also the message variant *#wxScrollWin{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxScrollWinEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`getOrientation(This::wxScrollWinEvent()) -> integer()`

See **external documentation**.

`getPosition(This::wxScrollWinEvent()) -> integer()`

See **external documentation**.

## wxScrolledWindow

---

Erlang module

See external documentation: **wxScrolledWindow**.

This class is derived (and can use functions) from:

*wxPanel*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxScrolledWindow()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxScrolledWindow()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxScrolledWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxScrolledWindow()`

Types:

**Option** = {winid, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`calcScrolledPosition(This::wxScrolledWindow(), Pt::{X::integer(), Y::integer()}) -> {X::integer(), Y::integer()}`

See external documentation.

`calcScrolledPosition(This::wxScrolledWindow(), X::integer(), Y::integer()) -> {Xx::integer(), Yy::integer()}`

See external documentation.

`calcUnscrolledPosition(This::wxScrolledWindow(), Pt::{X::integer(), Y::integer()}) -> {X::integer(), Y::integer()}`

See external documentation.

```
calcUnscrolledPosition(This::wxScrolledWindow(), X::integer(), Y::integer())  
-> {Xx::integer(), Yy::integer()}
```

See external documentation.

```
enableScrolling(This::wxScrolledWindow(), X_scrolling::bool(),  
Y_scrolling::bool()) -> ok
```

See external documentation.

```
getScrollPixelsPerUnit(This::wxScrolledWindow()) ->  
{PixelsPerUnitX::integer(), PixelsPerUnitY::integer()}
```

See external documentation.

```
getViewStart(This::wxScrolledWindow()) -> {X::integer(), Y::integer()}
```

See external documentation.

```
doPrepareDC(This::wxScrolledWindow(), Dc::wxDC() (see module wxDC)) -> ok
```

See external documentation.

```
prepareDC(This::wxScrolledWindow(), Dc::wxDC() (see module wxDC)) -> ok
```

See external documentation.

```
scroll(This::wxScrolledWindow(), X::integer(), Y::integer()) -> ok
```

See external documentation.

```
setScrollbars(This::wxScrolledWindow(), PixelsPerUnitX::integer(),  
PixelsPerUnitY::integer(), NoUnitsX::integer(), NoUnitsY::integer()) -> ok
```

Equivalent to *setScrollbars(This, PixelsPerUnitX, PixelsPerUnitY, NoUnitsX, NoUnitsY, [])*.

```
setScrollbars(This::wxScrolledWindow(), PixelsPerUnitX::integer(),  
PixelsPerUnitY::integer(), NoUnitsX::integer(), NoUnitsY::integer(),  
Options::[Option]) -> ok
```

Types:

**Option** = {xPos, integer()} | {yPos, integer()} | {noRefresh, bool()}

See external documentation.

```
setScrollRate(This::wxScrolledWindow(), Xstep::integer(), Ystep::integer()) -  
> ok
```

See external documentation.

```
setTargetWindow(This::wxScrolledWindow(), Target::wxWindow() (see module  
wxWindow)) -> ok
```

See external documentation.

## **wxScrolledWindow**

---

**destroy(This::wxScrolledWindow()) -> ok**

Destroys this object, do not use object again

## wxSetCursorEvent

---

Erlang module

See external documentation: **wxSetCursorEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*set\_cursor*

See also the message variant *#wxSetCursor{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

`wxSetCursorEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getCursor(This::wxSetCursorEvent()) -> wxCursor()` (see module `wxCursor`)

See external documentation.

`getX(This::wxSetCursorEvent()) -> integer()`

See external documentation.

`getY(This::wxSetCursorEvent()) -> integer()`

See external documentation.

`hasCursor(This::wxSetCursorEvent()) -> bool()`

See external documentation.

`setCursor(This::wxSetCursorEvent(), Cursor::wxCursor())` (see module `wxCursor`)  
`-> ok`

See external documentation.

## wxShowEvent

---

Erlang module

See external documentation: **wxShowEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*show*

See also the message variant *#wxShow{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

`wxShowEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

**`setShow(This::wxShowEvent(), Show::bool()) -> ok`**

See **external documentation**.

**`getShow(This::wxShowEvent()) -> bool()`**

See **external documentation**.



# wxSingleChoiceDialog

Erlang module

See external documentation: **wxSingleChoiceDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxSingleChoiceDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxSingleChoiceDialog()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Message::string(),  
Caption::string(), Choices::[[string()]]) -> wxSingleChoiceDialog()`

Equivalent to `new(Parent, Message, Caption, Choices, [])`.

`new(Parent::wxWindow() (see module wxWindow), Message::string(),  
Caption::string(), Choices::[[string()]], Options::[Option]) ->  
wxSingleChoiceDialog()`

Types:

`Option = {style, integer()} | {pos, {X::integer(), Y::integer()}}`

See external documentation.

`getSelection(This::wxSingleChoiceDialog()) -> integer()`

See external documentation.

`getStringSelection(This::wxSingleChoiceDialog()) -> string()`

See external documentation.

`setSelection(This::wxSingleChoiceDialog(), Sel::integer()) -> ok`

See external documentation.

`destroy(This::wxSingleChoiceDialog()) -> ok`

Destroys this object, do not use object again

## wxSizeEvent

---

Erlang module

See external documentation: **wxSizeEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*size*

See also the message variant *#wxSize{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

## DATA TYPES

`wxSizeEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

`getSize(This::wxSizeEvent()) -> {W::integer(), H::integer()}`

See **external documentation**.

## wxSizer

---

Erlang module

See external documentation: **wxSizer**.

### DATA TYPES

`wxSizer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`add(This::wxSizer(), Window::wxWindow() (see module wxWindow) | wxSizer()) -> wxSizerItem() (see module wxSizerItem)`**

Equivalent to *add(This, Window, [])*.

**`add(This::wxSizer(), X::integer() | term(), X::integer() | term()) -> wxSizerItem() (see module wxSizerItem)`**

See external documentation.

Alternatives:

**`add(This::wxSizer(), Width::integer(), Height::integer()) -> add(This, Width, Height, [])`**

**`add(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(), [Option]) -> wxSizerItem:wxSizerItem()`**

**Option** = {proportion, integer()} | {flag, integer()} | {border, integer()} | {userData, wx:wx() }

**`add(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(), Flags::wxSizerFlags:wxSizerFlags()) -> wxSizerItem:wxSizerItem()`**

**`add(This::wxSizer(), Width::integer(), Height::integer(), Options::[Option]) -> wxSizerItem() (see module wxSizerItem)`**

Types:

**Option** = {proportion, integer()} | {flag, integer()} | {border, integer()} | {userData, wx() (see module wx)}

See external documentation.

**`addSpacer(This::wxSizer(), Size::integer()) -> wxSizerItem() (see module wxSizerItem)`**

See external documentation.

**`addStretchSpacer(This::wxSizer()) -> wxSizerItem() (see module wxSizerItem)`**

Equivalent to *addStretchSpacer(This, [])*.

`addStretchSpacer(This::wxSizer(), Options::[Option]) -> wxSizerItem()` (see module `wxSizerItem`)

Types:

**Option** = {prop, integer()}

See external documentation.

`calcMin(This::wxSizer()) -> {W::integer(), H::integer()}`

See external documentation.

`clear(This::wxSizer()) -> ok`

Equivalent to `clear(This, [])`.

`clear(This::wxSizer(), Options::[Option]) -> ok`

Types:

**Option** = {delete\_windows, bool()}

See external documentation.

`detach(This::wxSizer(), X::integer() | term()) -> bool()`

See external documentation.

Alternatives:

`detach(This::wxSizer(), Index::integer()) -> bool()`

`detach(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer()) -> bool()`

`fit(This::wxSizer(), Window::wxWindow() (see module wxWindow)) -> {W::integer(), H::integer()}`

See external documentation.

`fitInside(This::wxSizer(), Window::wxWindow() (see module wxWindow)) -> ok`

See external documentation.

`getChildren(This::wxSizer()) -> [wxSizerItem() (see module wxSizerItem)]`

See external documentation.

`getItem(This::wxSizer(), X::term() | integer()) -> wxSizerItem()` (see module `wxSizerItem`)

See external documentation.

Alternatives:

`getItem(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer()) -> getItem(This, Window, [])`

`getItem(This::wxSizer(), Index::integer()) -> wxSizerItem:wxSizerItem()`

```
getItem(This::wxSizer(), Window::wxWindow() (see module wxWindow) |  
wxSizer(), Options::[Option]) -> wxSizerItem() (see module wxSizerItem)
```

Types:

**Option** = {recursive, bool()}

See external documentation.

```
getSize(This::wxSizer()) -> {W::integer(), H::integer()}
```

See external documentation.

```
getPosition(This::wxSizer()) -> {X::integer(), Y::integer()}
```

See external documentation.

```
getMinSize(This::wxSizer()) -> {W::integer(), H::integer()}
```

See external documentation.

```
hide(This::wxSizer(), X::term() | integer()) -> bool()
```

See external documentation.

Alternatives:

```
hide(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer()) ->  
hide(This, Window, [])
```

```
hide(This::wxSizer(), Index::integer()) -> bool()
```

```
hide(This::wxSizer(), Window::wxWindow() (see module wxWindow) | wxSizer(),  
Options::[Option]) -> bool()
```

Types:

**Option** = {recursive, bool()}

See external documentation.

```
insert(This::wxSizer(), Index::integer(), X::term()) -> wxSizerItem() (see  
module wxSizerItem)
```

See external documentation.

Alternatives:

```
insert(This::wxSizer(), Index::integer(), Window::wxWindow:wxWindow() |  
wxSizer()) -> insert(This, Index, Window, [])
```

```
insert(This::wxSizer(), Index::integer(), Item::wxSizerItem:wxSizerItem()) -  
> wxSizerItem:wxSizerItem()
```

```
insert(This::wxSizer(), Index::integer(), X::integer() | term(), X::integer()  
| term()) -> wxSizerItem() (see module wxSizerItem)
```

See external documentation.

Alternatives:

```
insert(This::wxSizer(), Index::integer(), Width::integer(), Height::integer())  
-> insert(This, Index, Width, Height, [])
```

```
insert(This::wxSizer(), Index::integer(), Window::wxWindow:wxWindow() |  
wxSizer(), [Option]) -> wxSizerItem:wxSizerItem()  
Option = {proportion, integer()} | {flag, integer()} | {border, integer()} | {userData, wx:wx()}  
  
insert(This::wxSizer(), Index::integer(), Window::wxWindow:wxWindow() |  
wxSizer(), Flags::wxSizerFlags:wxSizerFlags()) -> wxSizerItem:wxSizerItem()
```

```
insert(This::wxSizer(), Index::integer(), Width::integer(),  
Height::integer(), Options::[Option]) -> wxSizerItem() (see module  
wxSizerItem)
```

Types:

**Option** = {proportion, integer()} | {flag, integer()} | {border, integer()} | {userData, wx()} (see module wx)

See [external documentation](#).

```
insertSpacer(This::wxSizer(), Index::integer(), Size::integer()) ->  
wxSizerItem() (see module wxSizerItem)
```

See [external documentation](#).

```
insertStretchSpacer(This::wxSizer(), Index::integer()) -> wxSizerItem() (see  
module wxSizerItem)
```

Equivalent to *insertStretchSpacer(This, Index, [])*.

```
insertStretchSpacer(This::wxSizer(), Index::integer(), Options::[Option]) ->  
wxSizerItem() (see module wxSizerItem)
```

Types:

**Option** = {prop, integer()}

See [external documentation](#).

```
isShown(This::wxSizer(), X::integer() | term()) -> bool()
```

See [external documentation](#).

Alternatives:

```
isShown(This::wxSizer(), Index::integer()) -> bool()
```

```
isShown(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer()) -> bool()
```

```
layout(This::wxSizer()) -> ok
```

See [external documentation](#).

```
prepend(This::wxSizer(), X::term()) -> wxSizerItem() (see module wxSizerItem)
```

See [external documentation](#).

Alternatives:

```
prepend(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer()) ->  
prepend(This, Window, [])
```

```
prepend(This::wxSizer(), Item::wxSizerItem:wxSizerItem()) ->  
wxSizerItem:wxSizerItem()
```

```
prepend(This::wxSizer(), X::integer() | term(), X::integer() | term()) ->  
wxSizerItem() (see module wxSizerItem)
```

See external documentation.

Alternatives:

```
prepend(This::wxSizer(), Width::integer(), Height::integer()) ->  
prepend(This,Width,Height, [])
```

```
prepend(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(), [Option]) -  
> wxSizerItem:wxSizerItem()
```

**Option = {proportion, integer()} | {flag, integer()} | {border, integer()} | {userData, wx:wx() }**

```
prepend(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(),  
Flags::wxSizerFlags:wxSizerFlags()) -> wxSizerItem:wxSizerItem()
```

```
prepend(This::wxSizer(), Width::integer(), Height::integer(), Options::  
[Option]) -> wxSizerItem() (see module wxSizerItem)
```

Types:

**Option = {proportion, integer()} | {flag, integer()} | {border, integer()} | {userData, wx() (see module wx)}**

See external documentation.

```
prependSpacer(This::wxSizer(), Size::integer()) -> wxSizerItem() (see module  
wxSizerItem)
```

See external documentation.

```
prependStretchSpacer(This::wxSizer()) -> wxSizerItem() (see module  
wxSizerItem)
```

Equivalent to *prependStretchSpacer(This, [])*.

```
prependStretchSpacer(This::wxSizer(), Options::[Option]) -> wxSizerItem()  
(see module wxSizerItem)
```

Types:

**Option = {prop, integer() }**

See external documentation.

```
recalcSizes(This::wxSizer()) -> ok
```

See external documentation.

```
remove(This::wxSizer(), X::integer() | wxSizer()) -> bool()
```

See external documentation.

Alternatives:

```
remove(This::wxSizer(), Index::integer()) -> bool()
```

```
remove(This::wxSizer(), Sizer::wxSizer()) -> bool()
```

```
replace(This::wxSizer(), X::term() | integer(), X::term()) -> bool()
```

See external documentation.

Alternatives:

```
replace(This::wxSizer(), Oldwin::wxWindow:wxWindow() | wxSizer(),  
Newwin::wxWindow:wxWindow() | wxSizer()) -> replace(This,Oldwin,Newwin, [])
```

```
replace(This::wxSizer(), Index::integer(),  
Newitem::wxSizerItem:wxSizerItem()) -> bool()
```

```
replace(This::wxSizer(), Oldwin::wxWindow() (see module wxWindow) |  
wxSizer(), Newwin::wxWindow() (see module wxWindow) | wxSizer(), Options::  
[Option]) -> bool()
```

Types:

**Option** = {recursive, bool()}

See [external documentation](#).

```
setDimension(This::wxSizer(), X::integer(), Y::integer(), Width::integer(),  
Height::integer()) -> ok
```

See [external documentation](#).

```
setMinSize(This::wxSizer(), Size::{W::integer(), H::integer()}) -> ok
```

See [external documentation](#).

```
setMinSize(This::wxSizer(), Width::integer(), Height::integer()) -> ok
```

See [external documentation](#).

```
setItemMinSize(This::wxSizer(), X::integer() | term(), Size::{W::integer(),  
H::integer()}) -> bool()
```

See [external documentation](#).

Alternatives:

```
setItemMinSize(This::wxSizer(), Index::integer(), Size::  
{W::integer(),H::integer()}) -> bool()
```

```
setItemMinSize(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(),  
Size::{W::integer(),H::integer()}) -> bool()
```

```
setItemMinSize(This::wxSizer(), X::integer() | term(), Width::integer(),  
Height::integer()) -> bool()
```

See [external documentation](#).

Alternatives:

```
setItemMinSize(This::wxSizer(), Index::integer(), Width::integer(),  
Height::integer()) -> bool()
```

```
setItemMinSize(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(),  
Width::integer(), Height::integer()) -> bool()
```

```
setSizeHints(This::wxSizer(), Window::wxWindow() (see module wxWindow)) -> ok
```

See [external documentation](#).



**setVirtualSizeHints(This::wxSizer(), Window::wxWindow() (see module wxWindow)) -> ok**

See external documentation.

**show(This::wxSizer(), X::integer() | term() | bool()) -> bool() | bool() | ok**

See external documentation.

Alternatives:

**show(This::wxSizer(), Index::integer()) -> show(This, Index, [])**

**show(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer()) -> show(This, Window, [])**

**show(This::wxSizer(), Show::bool()) -> ok**

**show(This::wxSizer(), X::integer() | term(), Options::[Option]) -> bool()**

See external documentation.

Alternatives:

**show(This::wxSizer(), Index::integer(), [Option]) -> bool()**

Option = {show, bool()}

**show(This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(), [Option]) -> bool()**

Option = {show, bool()} | {recursive, bool()}

## wxSizerFlags

---

Erlang module

See external documentation: **wxSizerFlags**.

### DATA TYPES

`wxSizerFlags()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()`** -> **`wxSizerFlags()`**

Equivalent to `new([])`.

**`new(Options::[Option])`** -> **`wxSizerFlags()`**

Types:

**Option** = {proportion, integer()}

See external documentation.

**`align(This::wxSizerFlags(), Alignment::integer())`** -> **`wxSizerFlags()`**

See external documentation.

**`border(This::wxSizerFlags())`** -> **`wxSizerFlags()`**

Equivalent to `border(This, [])`.

**`border(This::wxSizerFlags(), Options::[Option])`** -> **`wxSizerFlags()`**

Types:

**Option** = {direction, integer()}

See external documentation.

**`border(This::wxSizerFlags(), Direction::integer(), BorderInPixels::integer())`**  
-> **`wxSizerFlags()`**

See external documentation.

**`center(This::wxSizerFlags())`** -> **`wxSizerFlags()`**

See external documentation.

**`centre(This::wxSizerFlags())`** -> **`wxSizerFlags()`**

See external documentation.

**expand(This::wxSizerFlags()) -> wxSizerFlags()**

See external documentation.

**left(This::wxSizerFlags()) -> wxSizerFlags()**

See external documentation.

**proportion(This::wxSizerFlags(), Proportion::integer()) -> wxSizerFlags()**

See external documentation.

**right(This::wxSizerFlags()) -> wxSizerFlags()**

See external documentation.

**destroy(This::wxSizerFlags()) -> ok**

Destroys this object, do not use object again

## wxSizerItem

---

Erlang module

See external documentation: **wxSizerItem**.

### DATA TYPES

`wxSizerItem()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxSizerItem()`

See external documentation.

`new(Window::wxWindow() (see module wxWindow) | wxSizer() (see module wxSizer), Flags::wxSizerFlags() (see module wxSizerFlags)) -> wxSizerItem()`

See external documentation.

`new(Width::integer(), Height::integer(), Flags::wxSizerFlags() (see module wxSizerFlags)) -> wxSizerItem()`

See external documentation.

`new(Window::wxWindow() (see module wxWindow) | wxSizer() (see module wxSizer), Proportion::integer(), Flag::integer(), Border::integer(), UserData::wx() (see module wx)) -> wxSizerItem()`

See external documentation.

`new(Width::integer(), Height::integer(), Proportion::integer(), Flag::integer(), Border::integer(), UserData::wx() (see module wx)) -> wxSizerItem()`

See external documentation.

`calcMin(This::wxSizerItem()) -> {W::integer(), H::integer()}`

See external documentation.

`deleteWindows(This::wxSizerItem()) -> ok`

See external documentation.

`detachSizer(This::wxSizerItem()) -> ok`

See external documentation.

`getBorder(This::wxSizerItem()) -> integer()`

See external documentation.

`getFlag(This::wxSizerItem()) -> integer()`

See external documentation.

`getMinSize(This::wxSizerItem()) -> {W::integer(), H::integer()}`

See external documentation.

`getPosition(This::wxSizerItem()) -> {X::integer(), Y::integer()}`

See external documentation.

`getProportion(This::wxSizerItem()) -> integer()`

See external documentation.

`getRatio(This::wxSizerItem()) -> float()`

See external documentation.

`getRect(This::wxSizerItem()) -> {X::integer(), Y::integer(), W::integer(),  
H::integer()}`

See external documentation.

`getSize(This::wxSizerItem()) -> {W::integer(), H::integer()}`

See external documentation.

`getSizer(This::wxSizerItem()) -> wxSizer() (see module wxSizer)`

See external documentation.

`getSpacer(This::wxSizerItem()) -> {W::integer(), H::integer()}`

See external documentation.

`getUserData(This::wxSizerItem()) -> wx() (see module wx)`

See external documentation.

`getWindow(This::wxSizerItem()) -> wxWindow() (see module wxWindow)`

See external documentation.

`isSizer(This::wxSizerItem()) -> bool()`

See external documentation.

`isShown(This::wxSizerItem()) -> bool()`

See external documentation.

`isSpacer(This::wxSizerItem()) -> bool()`

See external documentation.

`isWindow(This::wxSizerItem()) -> bool()`

See external documentation.

`setBorder(This::wxSizerItem(), Border::integer()) -> ok`

See external documentation.

`setDimension(This::wxSizerItem(), Pos::{X::integer(), Y::integer()}, Size::{W::integer(), H::integer()}) -> ok`

See external documentation.

`setFlag(This::wxSizerItem(), Flag::integer()) -> ok`

See external documentation.

`setInitSize(This::wxSizerItem(), X::integer(), Y::integer()) -> ok`

See external documentation.

`setMinSize(This::wxSizerItem(), Size::{W::integer(), H::integer()}) -> ok`

See external documentation.

`setMinSize(This::wxSizerItem(), X::integer(), Y::integer()) -> ok`

See external documentation.

`setProportion(This::wxSizerItem(), Proportion::integer()) -> ok`

See external documentation.

`setRatio(This::wxSizerItem(), X::float() | term()) -> ok`

See external documentation.

Alternatives:

`setRatio(This::wxSizerItem(), Ratio::float()) -> ok`

`setRatio(This::wxSizerItem(), Size::{W::integer(), H::integer()}) -> ok`

`setRatio(This::wxSizerItem(), Width::integer(), Height::integer()) -> ok`

See external documentation.

`setSizer(This::wxSizerItem(), Sizer::wxSizer() (see module wxSizer)) -> ok`

See external documentation.

`setSpacer(This::wxSizerItem(), Size::{W::integer(), H::integer()}) -> ok`

See external documentation.

**setSpacer(This::wxSizerItem(), Width::integer(), Height::integer()) -> ok**

See external documentation.

**setWindow(This::wxSizerItem(), Window::wxWindow() (see module wxWindow)) -> ok**

See external documentation.

**show(This::wxSizerItem(), Show::bool()) -> ok**

See external documentation.

**destroy(This::wxSizerItem()) -> ok**

Destroys this object, do not use object again

## wxSlider

---

Erlang module

See external documentation: **wxSlider**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxSlider()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxSlider()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Value::integer(), MinValue::integer(), MaxValue::integer()) -> wxSlider()`

Equivalent to `new(Parent, Id, Value, MinValue, MaxValue, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Value::integer(), MinValue::integer(), MaxValue::integer(), Options::[Option]) -> wxSlider()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`create(This::wxSlider(), Parent::wxWindow() (see module wxWindow), Id::integer(), Value::integer(), MinValue::integer(), MaxValue::integer()) -> bool()`

Equivalent to `create(This, Parent, Id, Value, MinValue, MaxValue, [])`.

`create(This::wxSlider(), Parent::wxWindow() (see module wxWindow), Id::integer(), Value::integer(), MinValue::integer(), MaxValue::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.



**getLineSize(This::wxSlider()) -> integer()**

See external documentation.

**getMax(This::wxSlider()) -> integer()**

See external documentation.

**getMin(This::wxSlider()) -> integer()**

See external documentation.

**getPageSize(This::wxSlider()) -> integer()**

See external documentation.

**getThumbLength(This::wxSlider()) -> integer()**

See external documentation.

**getValue(This::wxSlider()) -> integer()**

See external documentation.

**setLineSize(This::wxSlider(), LineSize::integer()) -> ok**

See external documentation.

**setPageSize(This::wxSlider(), PageSize::integer()) -> ok**

See external documentation.

**setRange(This::wxSlider(), MinValue::integer(), MaxValue::integer()) -> ok**

See external documentation.

**setThumbLength(This::wxSlider(), LenPixels::integer()) -> ok**

See external documentation.

**setValue(This::wxSlider(), Value::integer()) -> ok**

See external documentation.

**destroy(This::wxSlider()) -> ok**

Destroys this object, do not use object again

## wxSpinButton

---

Erlang module

See external documentation: **wxSpinButton**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxSpinButton()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxSpinButton()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxSpinButton()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxSpinButton()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxSpinButton(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

`create(This::wxSpinButton(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`getMax(This::wxSpinButton()) -> integer()`

See external documentation.

**getMin(This::wxSpinButton()) -> integer()**

See external documentation.

**getValue(This::wxSpinButton()) -> integer()**

See external documentation.

**setRange(This::wxSpinButton(), MinVal::integer(), MaxVal::integer()) -> ok**

See external documentation.

**setValue(This::wxSpinButton(), Value::integer()) -> ok**

See external documentation.

**destroy(This::wxSpinButton()) -> ok**

Destroys this object, do not use object again

## wxSpinCtrl

---

Erlang module

See external documentation: **wxSpinCtrl**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxSpinCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxSpinCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxSpinCtrl()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxSpinCtrl()`

Types:

**Option** = {id, integer()} | {value, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {min, integer()} | {max, integer()} | {initial, integer()}

See external documentation.

`create(This::wxSpinCtrl(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

`create(This::wxSpinCtrl(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {id, integer()} | {value, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {min, integer()} | {max, integer()} | {initial, integer()}

See external documentation.

`setValue(This::wxSpinCtrl(), X::integer() | string()) -> ok`

See external documentation.

Alternatives:

`setValue(This::wxSpinCtrl(), Value::integer()) -> ok`

`setValue(This::wxSpinCtrl(), Text::string()) -> ok`

`getValue(This::wxSpinCtrl()) -> integer()`

See [external documentation](#).

`setRange(This::wxSpinCtrl(), MinVal::integer(), MaxVal::integer()) -> ok`

See [external documentation](#).

`setSelection(This::wxSpinCtrl(), From::integer(), To::integer()) -> ok`

See [external documentation](#).

`getMin(This::wxSpinCtrl()) -> integer()`

See [external documentation](#).

`getMax(This::wxSpinCtrl()) -> integer()`

See [external documentation](#).

`destroy(This::wxSpinCtrl()) -> ok`

Destroys this object, do not use object again

## wxSpinEvent

---

Erlang module

See external documentation: **wxSpinEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*command\_spinctrl\_updated, spin\_up, spin\_down, spin*

See also the message variant *#wxSpin{}* event record type.

This class is derived (and can use functions) from:

*wxNotifyEvent*

*wxCommandEvent*

*wxEvent*

## DATA TYPES

`wxSpinEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

`getPosition(This::wxSpinEvent()) -> integer()`

See **external documentation**.

`setPosition(This::wxSpinEvent(), Pos::integer()) -> ok`

See **external documentation**.

# wxSplashScreen

---

Erlang module

See external documentation: **wxSplashScreen**.

This class is derived (and can use functions) from:

*wxFrame*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxSplashScreen()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxSplashScreen()`

See external documentation.

`new(Bitmap::wxBitmap() (see module wxBitmap), SplashStyle::integer(),  
Milliseconds::integer(), Parent::wxWindow() (see module wxWindow),  
Id::integer()) -> wxSplashScreen()`

Equivalent to `new(Bitmap, SplashStyle, Milliseconds, Parent, Id, [])`.

`new(Bitmap::wxBitmap() (see module wxBitmap), SplashStyle::integer(),  
Milliseconds::integer(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Options::[Option]) -> wxSplashScreen()`

Types:

`Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}`

See external documentation.

`getSplashStyle(This::wxSplashScreen()) -> integer()`

See external documentation.

`getTimeout(This::wxSplashScreen()) -> integer()`

See external documentation.

`destroy(This::wxSplashScreen()) -> ok`

Destroys this object, do not use object again

## wxSplitterEvent

---

Erlang module

See external documentation: **wxSplitterEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*command\_splitter\_sash\_pos\_changed,* *command\_splitter\_sash\_pos\_changing,*  
*command\_splitter\_doubleclicked,* *command\_splitter\_unsplit*

See also the message variant *#wxSplitter{}* event record type.

This class is derived (and can use functions) from:

*wxNotifyEvent*

*wxCommandEvent*

*wxEvent*

## DATA TYPES

`wxSplitterEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getSashPosition(This::wxSplitterEvent()) -> integer()`

See external documentation.

`getX(This::wxSplitterEvent()) -> integer()`

See external documentation.

`getY(This::wxSplitterEvent()) -> integer()`

See external documentation.

`getWindowBeingRemoved(This::wxSplitterEvent()) -> wxWindow()` (see module `wxWindow`)

See external documentation.

`setSashPosition(This::wxSplitterEvent(), Pos::integer()) -> ok`

See external documentation.



# wxSplitterWindow

Erlang module

See external documentation: **wxSplitterWindow**.

This class is derived (and can use functions) from:

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxSplitterWindow()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxSplitterWindow()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxSplitterWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxSplitterWindow()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxSplitterWindow(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

`create(This::wxSplitterWindow(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`getMinimumPaneSize(This::wxSplitterWindow()) -> integer()`

See external documentation.

`getSashGravity(This::wxSplitterWindow()) -> float()`

See external documentation.

`getSashPosition(This::wxSplitterWindow()) -> integer()`

See external documentation.

`getSplitMode(This::wxSplitterWindow()) -> WxSplitMode`

Types:

`WxSplitMode = integer()`

See external documentation.

WxSplitMode is one of ?wxSPLIT\_HORIZONTAL | ?wxSPLIT\_VERTICAL

`getWindow1(This::wxSplitterWindow()) -> wxWindow() (see module wxWindow)`

See external documentation.

`getWindow2(This::wxSplitterWindow()) -> wxWindow() (see module wxWindow)`

See external documentation.

`initialize(This::wxSplitterWindow(), Window::wxWindow() (see module wxWindow)) -> ok`

See external documentation.

`isSplit(This::wxSplitterWindow()) -> bool()`

See external documentation.

`replaceWindow(This::wxSplitterWindow(), WinOld::wxWindow() (see module wxWindow), WinNew::wxWindow() (see module wxWindow)) -> bool()`

See external documentation.

`setSashGravity(This::wxSplitterWindow(), Gravity::float()) -> ok`

See external documentation.

`setSashPosition(This::wxSplitterWindow(), Position::integer()) -> ok`

Equivalent to `setSashPosition(This, Position, [])`.

`setSashPosition(This::wxSplitterWindow(), Position::integer(), Options::[Option]) -> ok`

Types:

`Option = {redraw, bool()}`

See external documentation.

`setSashSize(This::wxSplitterWindow(), Width::integer()) -> ok`

See external documentation.

**setMinimumPaneSize**(This::wxSplitterWindow(), Min::integer()) -> ok

See external documentation.

**setSplitMode**(This::wxSplitterWindow(), Mode::integer()) -> ok

See external documentation.

**splitHorizontally**(This::wxSplitterWindow(), Window1::wxWindow() (see module wxWindow), Window2::wxWindow() (see module wxWindow)) -> bool()

Equivalent to *splitHorizontally(This, Window1, Window2, [])*.

**splitHorizontally**(This::wxSplitterWindow(), Window1::wxWindow() (see module wxWindow), Window2::wxWindow() (see module wxWindow), Options::[Option]) -> bool()

Types:

**Option** = {sashPosition, integer()}

See external documentation.

**splitVertically**(This::wxSplitterWindow(), Window1::wxWindow() (see module wxWindow), Window2::wxWindow() (see module wxWindow)) -> bool()

Equivalent to *splitVertically(This, Window1, Window2, [])*.

**splitVertically**(This::wxSplitterWindow(), Window1::wxWindow() (see module wxWindow), Window2::wxWindow() (see module wxWindow), Options::[Option]) -> bool()

Types:

**Option** = {sashPosition, integer()}

See external documentation.

**unsplit**(This::wxSplitterWindow()) -> bool()

Equivalent to *unsplit(This, [])*.

**unsplit**(This::wxSplitterWindow(), Options::[Option]) -> bool()

Types:

**Option** = {toRemove, wxWindow() (see module wxWindow)}

See external documentation.

**updateSize**(This::wxSplitterWindow()) -> ok

See external documentation.

**destroy**(This::wxSplitterWindow()) -> ok

Destroys this object, do not use object again

## wxStaticBitmap

---

Erlang module

See external documentation: **wxStaticBitmap**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxStaticBitmap()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxStaticBitmap()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(),  
Label::wxBitmap() (see module wxBitmap)) -> wxStaticBitmap()`

Equivalent to `new(Parent, Id, Label, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(),  
Label::wxBitmap() (see module wxBitmap), Options::[Option]) ->  
wxStaticBitmap()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxStaticBitmap(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::wxBitmap() (see module wxBitmap)) -> bool()`

Equivalent to `create(This, Parent, Id, Label, [])`.

`create(This::wxStaticBitmap(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::wxBitmap() (see module wxBitmap), Options::[Option]) ->  
bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`getBitmap(This::wxStaticBitmap()) -> wxBitmap() (see module wxBitmap)`

See external documentation.

```
setBitmap(This::wxStaticBitmap(), Bitmap::wxBitmap() (see module wxBitmap)) -  
> ok
```

See external documentation.

```
destroy(This::wxStaticBitmap()) -> ok
```

Destroys this object, do not use object again

## wxStaticBox

---

Erlang module

See external documentation: **wxStaticBox**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxStaticBox()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxStaticBox()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string())  
-> wxStaticBox()`

Equivalent to `new(Parent, Id, Label, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string(),  
Options::[Option]) -> wxStaticBox()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxStaticBox(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string()) -> bool()`

Equivalent to `create(This, Parent, Id, Label, [])`.

`create(This::wxStaticBox(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`destroy(This::wxStaticBox()) -> ok`

Destroys this object, do not use object again

## wxStaticBoxSizer

---

Erlang module

See external documentation: **wxStaticBoxSizer**.

This class is derived (and can use functions) from:

*wxBoxSizer*

*wxSizer*

### DATA TYPES

`wxStaticBoxSizer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(X::integer() | term(), X::term() | integer()) -> wxStaticBoxSizer()`

See external documentation.

Alternatives:

`new(Orient::integer(), Win::wxWindow:wxWindow()) -> new(Orient,Win, [])`

`new(Box::wxStaticBox:wxStaticBox(), Orient::integer()) -> wxStaticBoxSizer()`

`new(Orient::integer(), Win::wxWindow() (see module wxWindow), Options::[Option]) -> wxStaticBoxSizer()`

Types:

**Option** = {label, string()}

See external documentation.

`getStaticBox(This::wxStaticBoxSizer()) -> wxStaticBox() (see module wxStaticBox)`

See external documentation.

`destroy(This::wxStaticBoxSizer()) -> ok`

Destroys this object, do not use object again

## wxStaticLine

---

Erlang module

See external documentation: **wxStaticLine**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxStaticLine()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxStaticLine()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxStaticLine()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxStaticLine()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxStaticLine(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

`create(This::wxStaticLine(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`isVertical(This::wxStaticLine()) -> bool()`

See external documentation.



**getDefaultSize()** -> integer()

See external documentation.

**destroy(This::wxStaticLine())** -> ok

Destroys this object, do not use object again

## wxStaticText

---

Erlang module

See external documentation: **wxStaticText**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxStaticText()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxStaticText()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string())  
-> wxStaticText()`

Equivalent to `new(Parent, Id, Label, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string(),  
Options::[Option]) -> wxStaticText()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxStaticText(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string()) -> bool()`

Equivalent to `create(This, Parent, Id, Label, [])`.

`create(This::wxStaticText(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`getLabel(This::wxStaticText()) -> string()`

See external documentation.

**setLabel(This::wxStaticText(), Label::string()) -> ok**

See external documentation.

**wrap(This::wxStaticText(), Width::integer()) -> ok**

See external documentation.

**destroy(This::wxStaticText()) -> ok**

Destroys this object, do not use object again

## wxStatusBar

---

Erlang module

See external documentation: **wxStatusBar**.

This class is derived (and can use functions) from:

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxStatusBar()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxStatusBar()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxStatusBar()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxStatusBar()`

Types:

**Option** = {winid, integer()} | {style, integer()}

See external documentation.

`create(This::wxStatusBar(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

`create(This::wxStatusBar(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {winid, integer()} | {style, integer()}

See external documentation.

`getFieldRect(This::wxStatusBar(), I::integer(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> bool()`

See external documentation.

`getFieldsCount(This::wxStatusBar()) -> integer()`

See external documentation.

```
getStatusText(This::wxStatusBar()) -> string()
```

Equivalent to *getStatusText(This, [])*.

```
getStatusText(This::wxStatusBar(), Options::[Option]) -> string()
```

Types:

**Option** = {number, integer()}

See **external documentation**.

```
popStatusText(This::wxStatusBar()) -> ok
```

Equivalent to *popStatusText(This, [])*.

```
popStatusText(This::wxStatusBar(), Options::[Option]) -> ok
```

Types:

**Option** = {number, integer()}

See **external documentation**.

```
pushStatusText(This::wxStatusBar(), Text::string()) -> ok
```

Equivalent to *pushStatusText(This, Text, [])*.

```
pushStatusText(This::wxStatusBar(), Text::string(), Options::[Option]) -> ok
```

Types:

**Option** = {number, integer()}

See **external documentation**.

```
setFieldsCount(This::wxStatusBar(), Number::integer()) -> ok
```

Equivalent to *setFieldsCount(This, Number, [])*.

```
setFieldsCount(This::wxStatusBar(), Number::integer(), Options::[Option]) ->  
ok
```

Types:

**Option** = {widths, [integer()]}

See **external documentation**.

```
setMinHeight(This::wxStatusBar(), Height::integer()) -> ok
```

See **external documentation**.

```
setStatusText(This::wxStatusBar(), Text::string()) -> ok
```

Equivalent to *setStatusText(This, Text, [])*.

```
setStatusText(This::wxStatusBar(), Text::string(), Options::[Option]) -> ok
```

Types:

**Option** = {number, integer()}

See **external documentation**.

## **wxStatusBar**

---

**setStatusWidths(This::wxStatusBar(), Widths\_field::[integer()]) -> ok**

See **external documentation**.

**setStatusStyles(This::wxStatusBar(), Styles::[integer()]) -> ok**

See **external documentation**.

**destroy(This::wxStatusBar()) -> ok**

Destroys this object, do not use object again

## wxStdDialogButtonSizer

---

Erlang module

See external documentation: **wxStdDialogButtonSizer**.

This class is derived (and can use functions) from:

*wxBoxSizer*

*wxSizer*

### DATA TYPES

`wxStdDialogButtonSizer()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxStdDialogButtonSizer()`

See external documentation.

`addButton(This::wxStdDialogButtonSizer(), Button::wxButton() (see module wxButton)) -> ok`

See external documentation.

`realize(This::wxStdDialogButtonSizer()) -> ok`

See external documentation.

`setAffirmativeButton(This::wxStdDialogButtonSizer(), Button::wxButton() (see module wxButton)) -> ok`

See external documentation.

`setCancelButton(This::wxStdDialogButtonSizer(), Button::wxButton() (see module wxButton)) -> ok`

See external documentation.

`setNegativeButton(This::wxStdDialogButtonSizer(), Button::wxButton() (see module wxButton)) -> ok`

See external documentation.

`destroy(This::wxStdDialogButtonSizer()) -> ok`

Destroys this object, do not use object again

## wxStyledTextCtrl

---

Erlang module

See external documentation: **wxStyledTextCtrl**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxStyledTextCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxStyledTextCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxStyledTextCtrl()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxStyledTextCtrl()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`create(This::wxStyledTextCtrl(), Parent::wxWindow() (see module wxWindow)) -> bool()`

Equivalent to `create(This, Parent, [])`.

`create(This::wxStyledTextCtrl(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()`

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`addText(This::wxStyledTextCtrl(), Text::string()) -> ok`

See external documentation.



`addStyledText(This::wxStyledTextCtrl(), Data::wxMemoryBuffer() (see module wxMemoryBuffer)) -> ok`

See external documentation.

`insertText(This::wxStyledTextCtrl(), Pos::integer(), Text::string()) -> ok`

See external documentation.

`clearAll(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`clearDocumentStyle(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`getLength(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`getCharAt(This::wxStyledTextCtrl(), Pos::integer()) -> integer()`

See external documentation.

`getCurrentPos(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`getAnchor(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`getStyleAt(This::wxStyledTextCtrl(), Pos::integer()) -> integer()`

See external documentation.

`redo(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setUndoCollection(This::wxStyledTextCtrl(), CollectUndo::bool()) -> ok`

See external documentation.

`selectAll(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setSavePoint(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`getStyledText(This::wxStyledTextCtrl(), startPos::integer(), EndPos::integer()) -> wxMemoryBuffer() (see module wxMemoryBuffer)`

See external documentation.

`canRedo(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`markerLineFromHandle(This::wxStyledTextCtrl(), Handle::integer()) -> integer()`

See external documentation.

`markerDeleteHandle(This::wxStyledTextCtrl(), Handle::integer()) -> ok`

See external documentation.

`getUndoCollection(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`getViewWhiteSpace(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setViewWhiteSpace(This::wxStyledTextCtrl(), ViewWS::integer()) -> ok`

See external documentation.

`positionFromPoint(This::wxStyledTextCtrl(), Pt::{X::integer(), Y::integer()}) -> integer()`

See external documentation.

`positionFromPointClose(This::wxStyledTextCtrl(), X::integer(), Y::integer()) -> integer()`

See external documentation.

`gotoLine(This::wxStyledTextCtrl(), Line::integer()) -> ok`

See external documentation.

`gotoPos(This::wxStyledTextCtrl(), Pos::integer()) -> ok`

See external documentation.

`setAnchor(This::wxStyledTextCtrl(), PosAnchor::integer()) -> ok`

See external documentation.

`getCurLine(This::wxStyledTextCtrl()) -> {string(), LinePos::integer()}`

See external documentation.

`getEndStyled(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`convertEOLs(This::wxStyledTextCtrl(), EolMode::integer()) -> ok`

See external documentation.

`getEOLMode(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setEOLMode(This::wxStyledTextCtrl(), EolMode::integer()) -> ok`

See external documentation.

`startStyling(This::wxStyledTextCtrl(), Pos::integer(), Mask::integer()) -> ok`

See external documentation.

`setStyling(This::wxStyledTextCtrl(), Length::integer(), Style::integer()) -> ok`

See external documentation.

`getBufferedDraw(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setBufferedDraw(This::wxStyledTextCtrl(), Buffered::bool()) -> ok`

See external documentation.

`setTabWidth(This::wxStyledTextCtrl(), TabWidth::integer()) -> ok`

See external documentation.

`getTabWidth(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setCodePage(This::wxStyledTextCtrl(), CodePage::integer()) -> ok`

See external documentation.

`markerDefine(This::wxStyledTextCtrl(), MarkerNumber::integer(), MarkerSymbol::integer()) -> ok`

Equivalent to *markerDefine(This, MarkerNumber, MarkerSymbol, [])*.

`markerDefine(This::wxStyledTextCtrl(), MarkerNumber::integer(), MarkerSymbol::integer(), Options::[Option]) -> ok`

Types:

**Option** = {foreground, colour() (see module wx)} | {background, colour() (see module wx)}

See external documentation.

`markerSetForeground(This::wxStyledTextCtrl(), MarkerNumber::integer(),  
Fore::colour() (see module wx)) -> ok`

See external documentation.

`markerSetBackground(This::wxStyledTextCtrl(), MarkerNumber::integer(),  
Back::colour() (see module wx)) -> ok`

See external documentation.

`markerAdd(This::wxStyledTextCtrl(), Line::integer(), MarkerNumber::integer())  
-> integer()`

See external documentation.

`markerDelete(This::wxStyledTextCtrl(), Line::integer(),  
MarkerNumber::integer()) -> ok`

See external documentation.

`markerDeleteAll(This::wxStyledTextCtrl(), MarkerNumber::integer()) -> ok`

See external documentation.

`markerGet(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`markerNext(This::wxStyledTextCtrl(), LineStart::integer(),  
MarkerMask::integer()) -> integer()`

See external documentation.

`markerPrevious(This::wxStyledTextCtrl(), LineStart::integer(),  
MarkerMask::integer()) -> integer()`

See external documentation.

`markerDefineBitmap(This::wxStyledTextCtrl(), MarkerNumber::integer(),  
Bmp::wxBitmap() (see module wxBitmap)) -> ok`

See external documentation.

`markerAddSet(This::wxStyledTextCtrl(), Line::integer(), Set::integer()) -> ok`

See external documentation.

`markerSetAlpha(This::wxStyledTextCtrl(), MarkerNumber::integer(),  
Alpha::integer()) -> ok`

See external documentation.

`setMarginType(This::wxStyledTextCtrl(), Margin::integer(),  
MarginType::integer()) -> ok`

See external documentation.

```
getMarginType(This::wxStyledTextCtrl(), Margin::integer()) -> integer()
```

See external documentation.

```
setMarginWidth(This::wxStyledTextCtrl(), Margin::integer(),  
PixelWidth::integer()) -> ok
```

See external documentation.

```
getMarginWidth(This::wxStyledTextCtrl(), Margin::integer()) -> integer()
```

See external documentation.

```
setMarginMask(This::wxStyledTextCtrl(), Margin::integer(), Mask::integer()) -  
> ok
```

See external documentation.

```
getMarginMask(This::wxStyledTextCtrl(), Margin::integer()) -> integer()
```

See external documentation.

```
setMarginSensitive(This::wxStyledTextCtrl(), Margin::integer(),  
Sensitive::bool()) -> ok
```

See external documentation.

```
getMarginSensitive(This::wxStyledTextCtrl(), Margin::integer()) -> bool()
```

See external documentation.

```
styleClearAll(This::wxStyledTextCtrl()) -> ok
```

See external documentation.

```
styleSetForeground(This::wxStyledTextCtrl(), Style::integer(), Fore::colour()  
(see module wx)) -> ok
```

See external documentation.

```
styleSetBackground(This::wxStyledTextCtrl(), Style::integer(), Back::colour()  
(see module wx)) -> ok
```

See external documentation.

```
styleSetBold(This::wxStyledTextCtrl(), Style::integer(), Bold::bool()) -> ok
```

See external documentation.

```
styleSetItalic(This::wxStyledTextCtrl(), Style::integer(), Italic::bool()) ->  
ok
```

See external documentation.

```
styleSetSize(This::wxStyledTextCtrl(), Style::integer(),  
SizePoints::integer()) -> ok
```

See external documentation.

```
styleSetFaceName(This::wxStyledTextCtrl(), Style::integer(),  
FontName::string()) -> ok
```

See external documentation.

```
styleSetEOLFilled(This::wxStyledTextCtrl(), Style::integer(), Filled::bool())  
-> ok
```

See external documentation.

```
styleResetDefault(This::wxStyledTextCtrl()) -> ok
```

See external documentation.

```
styleSetUnderline(This::wxStyledTextCtrl(), Style::integer(),  
Underline::bool()) -> ok
```

See external documentation.

```
styleSetCase(This::wxStyledTextCtrl(), Style::integer(),  
CaseForce::integer()) -> ok
```

See external documentation.

```
styleSetHotSpot(This::wxStyledTextCtrl(), Style::integer(), Hotspot::bool())  
-> ok
```

See external documentation.

```
setSelForeground(This::wxStyledTextCtrl(), UseSetting::bool(), Fore::colour()  
(see module wx)) -> ok
```

See external documentation.

```
setSelBackground(This::wxStyledTextCtrl(), UseSetting::bool(), Back::colour()  
(see module wx)) -> ok
```

See external documentation.

```
getSelAlpha(This::wxStyledTextCtrl()) -> integer()
```

See external documentation.

```
setSelAlpha(This::wxStyledTextCtrl(), Alpha::integer()) -> ok
```

See external documentation.

```
setCaretForeground(This::wxStyledTextCtrl(), Fore::colour() (see module wx))  
-> ok
```

See external documentation.

`cmdKeyAssign(This::wxStyledTextCtrl(), Key::integer(), Modifiers::integer(),  
Cmd::integer()) -> ok`

See external documentation.

`cmdKeyClear(This::wxStyledTextCtrl(), Key::integer(), Modifiers::integer()) -  
> ok`

See external documentation.

`cmdKeyClearAll(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setStyleBytes(This::wxStyledTextCtrl(), Length::integer()) -> integer()`

See external documentation.

`styleSetVisible(This::wxStyledTextCtrl(), Style::integer(), Visible::bool())  
-> ok`

See external documentation.

`getCaretPeriod(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setCaretPeriod(This::wxStyledTextCtrl(), PeriodMilliseconds::integer()) -> ok`

See external documentation.

`setWordChars(This::wxStyledTextCtrl(), Characters::string()) -> ok`

See external documentation.

`beginUndoAction(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`endUndoAction(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`indicatorSetStyle(This::wxStyledTextCtrl(), Indic::integer(),  
Style::integer()) -> ok`

See external documentation.

`indicatorGetStyle(This::wxStyledTextCtrl(), Indic::integer()) -> integer()`

See external documentation.

`indicatorSetForeground(This::wxStyledTextCtrl(), Indic::integer(),  
Fore::colour() (see module wx)) -> ok`

See external documentation.

`indicatorGetForeground(This::wxStyledTextCtrl(), Indic::integer()) -> colour()` (see module wx)

See external documentation.

`setWhitespaceForeground(This::wxStyledTextCtrl(), UseSetting::bool(), Fore::colour())` (see module wx) -> ok

See external documentation.

`setWhitespaceBackground(This::wxStyledTextCtrl(), UseSetting::bool(), Back::colour())` (see module wx) -> ok

See external documentation.

`getStyleBits(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setLineState(This::wxStyledTextCtrl(), Line::integer(), State::integer()) -> ok`

See external documentation.

`getLineState(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`getMaxLineState(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`getCaretLineVisible(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setCaretLineVisible(This::wxStyledTextCtrl(), Show::bool()) -> ok`

See external documentation.

`getCaretLineBackground(This::wxStyledTextCtrl()) -> colour()` (see module wx)

See external documentation.

`setCaretLineBackground(This::wxStyledTextCtrl(), Back::colour())` (see module wx) -> ok

See external documentation.

`autoCompShow(This::wxStyledTextCtrl(), LenEntered::integer(), ItemList::string()) -> ok`

See external documentation.

`autoCompCancel(This::wxStyledTextCtrl()) -> ok`

See external documentation.



`autoCompActive(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`autoCompPosStart(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`autoCompComplete(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`autoCompStops(This::wxStyledTextCtrl(), CharacterSet::string()) -> ok`

See external documentation.

`autoCompSetSeparator(This::wxStyledTextCtrl(), SeparatorCharacter::integer())  
-> ok`

See external documentation.

`autoCompGetSeparator(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`autoCompSelect(This::wxStyledTextCtrl(), Text::string()) -> ok`

See external documentation.

`autoCompSetCancelAtStart(This::wxStyledTextCtrl(), Cancel::bool()) -> ok`

See external documentation.

`autoCompGetCancelAtStart(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`autoCompSetFillUps(This::wxStyledTextCtrl(), CharacterSet::string()) -> ok`

See external documentation.

`autoCompSetChooseSingle(This::wxStyledTextCtrl(), ChooseSingle::bool()) -> ok`

See external documentation.

`autoCompGetChooseSingle(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`autoCompSetIgnoreCase(This::wxStyledTextCtrl(), IgnoreCase::bool()) -> ok`

See external documentation.

`autoCompGetIgnoreCase(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

```
userListShow(This::wxStyledTextCtrl(), ListType::integer(),  
ItemList::string()) -> ok
```

See external documentation.

```
autoCompSetAutoHide(This::wxStyledTextCtrl(), AutoHide::bool()) -> ok
```

See external documentation.

```
autoCompGetAutoHide(This::wxStyledTextCtrl()) -> bool()
```

See external documentation.

```
autoCompSetDropRestOfWord(This::wxStyledTextCtrl(), DropRestOfWord::bool()) -  
> ok
```

See external documentation.

```
autoCompGetDropRestOfWord(This::wxStyledTextCtrl()) -> bool()
```

See external documentation.

```
registerImage(This::wxStyledTextCtrl(), Type::integer(), Bmp::wxBitmap() (see  
module wxBitmap)) -> ok
```

See external documentation.

```
clearRegisteredImages(This::wxStyledTextCtrl()) -> ok
```

See external documentation.

```
autoCompGetTypeSeparator(This::wxStyledTextCtrl()) -> integer()
```

See external documentation.

```
autoCompSetTypeSeparator(This::wxStyledTextCtrl(),  
SeparatorCharacter::integer()) -> ok
```

See external documentation.

```
autoCompSetMaxWidth(This::wxStyledTextCtrl(), CharacterCount::integer()) ->  
ok
```

See external documentation.

```
autoCompGetMaxWidth(This::wxStyledTextCtrl()) -> integer()
```

See external documentation.

```
autoCompSetMaxHeight(This::wxStyledTextCtrl(), RowCount::integer()) -> ok
```

See external documentation.

```
autoCompGetMaxHeight(This::wxStyledTextCtrl()) -> integer()
```

See external documentation.

`setIndent(This::wxStyledTextCtrl(), IndentSize::integer()) -> ok`

See external documentation.

`getIndent(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setUseTabs(This::wxStyledTextCtrl(), UseTabs::bool()) -> ok`

See external documentation.

`getUseTabs(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setLineIndentation(This::wxStyledTextCtrl(), Line::integer(),  
IndentSize::integer()) -> ok`

See external documentation.

`getLineIndentation(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`getLineIndentPosition(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`getColumn(This::wxStyledTextCtrl(), Pos::integer()) -> integer()`

See external documentation.

`setUseHorizontalScrollBar(This::wxStyledTextCtrl(), Show::bool()) -> ok`

See external documentation.

`getUseHorizontalScrollBar(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setIndentationGuides(This::wxStyledTextCtrl(), Show::bool()) -> ok`

See external documentation.

`getIndentationGuides(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setHighlightGuide(This::wxStyledTextCtrl(), Column::integer()) -> ok`

See external documentation.

`getHighlightGuide(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`getLineEndPosition(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`getCodePage(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`getCaretForeground(This::wxStyledTextCtrl()) -> colour() (see module wx)`

See external documentation.

`getReadOnly(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setCurrentPos(This::wxStyledTextCtrl(), Pos::integer()) -> ok`

See external documentation.

`setSelectionStart(This::wxStyledTextCtrl(), Pos::integer()) -> ok`

See external documentation.

`getSelectionStart(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setSelectionEnd(This::wxStyledTextCtrl(), Pos::integer()) -> ok`

See external documentation.

`getSelectionEnd(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setPrintMagnification(This::wxStyledTextCtrl(), Magnification::integer()) -> ok`

See external documentation.

`getPrintMagnification(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setPrintColourMode(This::wxStyledTextCtrl(), Mode::integer()) -> ok`

See external documentation.

`getPrintColourMode(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`findText(This::wxStyledTextCtrl(), MinPos::integer(), MaxPos::integer(), Text::string()) -> integer()`

Equivalent to *findText(This, MinPos, MaxPos, Text, [])*.

```
findText(This::wxStyledTextCtrl(), MinPos::integer(), MaxPos::integer(),  
Text::string(), Options::[Option]) -> integer()
```

Types:

Option = {flags, integer()}

See external documentation.

```
formatRange(This::wxStyledTextCtrl(), DoDraw::bool(), StartPos::integer(),  
EndPos::integer(), Draw::wxDC() (see module wxDC), Target::wxDC() (see module  
wxDC), RenderRect::{X::integer(), Y::integer(), W::integer(), H::integer()},  
PageRect::{X::integer(), Y::integer(), W::integer(), H::integer()}) ->  
integer()
```

See external documentation.

```
getFirstVisibleLine(This::wxStyledTextCtrl()) -> integer()
```

See external documentation.

```
getLine(This::wxStyledTextCtrl(), Line::integer()) -> string()
```

See external documentation.

```
getLineCount(This::wxStyledTextCtrl()) -> integer()
```

See external documentation.

```
setMarginLeft(This::wxStyledTextCtrl(), PixelWidth::integer()) -> ok
```

See external documentation.

```
getMarginLeft(This::wxStyledTextCtrl()) -> integer()
```

See external documentation.

```
setMarginRight(This::wxStyledTextCtrl(), PixelWidth::integer()) -> ok
```

See external documentation.

```
getMarginRight(This::wxStyledTextCtrl()) -> integer()
```

See external documentation.

```
getModify(This::wxStyledTextCtrl()) -> bool()
```

See external documentation.

```
setSelection(This::wxStyledTextCtrl(), Start::integer(), End::integer()) ->  
ok
```

See external documentation.

```
getSelectedText(This::wxStyledTextCtrl()) -> string()
```

See external documentation.

`getTextRange(This::wxStyledTextCtrl(), startPos::integer(),  
endPos::integer()) -> string()`

See external documentation.

`hideSelection(This::wxStyledTextCtrl(), Normal::bool()) -> ok`

See external documentation.

`lineFromPosition(This::wxStyledTextCtrl(), Pos::integer()) -> integer()`

See external documentation.

`positionFromLine(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`lineScroll(This::wxStyledTextCtrl(), Columns::integer(), Lines::integer()) ->  
ok`

See external documentation.

`ensureCaretVisible(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`replaceSelection(This::wxStyledTextCtrl(), Text::string()) -> ok`

See external documentation.

`setReadOnly(This::wxStyledTextCtrl(), ReadOnly::bool()) -> ok`

See external documentation.

`canPaste(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`canUndo(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`emptyUndoBuffer(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`undo(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`cut(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`copy(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`paste(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`clear(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setText(This::wxStyledTextCtrl(), Text::string()) -> ok`

See external documentation.

`getText(This::wxStyledTextCtrl()) -> string()`

See external documentation.

`getLength(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`getOvertype(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setCaretWidth(This::wxStyledTextCtrl(), PixelWidth::integer()) -> ok`

See external documentation.

`getCaretWidth(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setTargetStart(This::wxStyledTextCtrl(), Pos::integer()) -> ok`

See external documentation.

`getTargetStart(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setTargetEnd(This::wxStyledTextCtrl(), Pos::integer()) -> ok`

See external documentation.

`getTargetEnd(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`replaceTarget(This::wxStyledTextCtrl(), Text::string()) -> integer()`

See external documentation.

`searchInTarget(This::wxStyledTextCtrl(), Text::string()) -> integer()`

See external documentation.

`setSearchFlags(This::wxStyledTextCtrl(), Flags::integer()) -> ok`

See external documentation.

`getSearchFlags(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`callTipShow(This::wxStyledTextCtrl(), Pos::integer(), Definition::string()) -> ok`

See external documentation.

`callTipCancel(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`callTipActive(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`callTipPosAtStart(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`callTipSetHighlight(This::wxStyledTextCtrl(), Start::integer(), End::integer()) -> ok`

See external documentation.

`callTipSetBackground(This::wxStyledTextCtrl(), Back::colour() (see module wx)) -> ok`

See external documentation.

`callTipSetForeground(This::wxStyledTextCtrl(), Fore::colour() (see module wx)) -> ok`

See external documentation.

`callTipSetForegroundHighlight(This::wxStyledTextCtrl(), Fore::colour() (see module wx)) -> ok`

See external documentation.

`callTipUseStyle(This::wxStyledTextCtrl(), TabSize::integer()) -> ok`

See external documentation.

`visibleFromDocLine(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`docLineFromVisible(This::wxStyledTextCtrl(), LineDisplay::integer()) -> integer()`

See external documentation.



`wrapCount(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`setFoldLevel(This::wxStyledTextCtrl(), Line::integer(), Level::integer()) -> ok`

See external documentation.

`getFoldLevel(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`getLastChild(This::wxStyledTextCtrl(), Line::integer(), Level::integer()) -> integer()`

See external documentation.

`getFoldParent(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`showLines(This::wxStyledTextCtrl(), LineStart::integer(), LineEnd::integer()) -> ok`

See external documentation.

`hideLines(This::wxStyledTextCtrl(), LineStart::integer(), LineEnd::integer()) -> ok`

See external documentation.

`getLineVisible(This::wxStyledTextCtrl(), Line::integer()) -> bool()`

See external documentation.

`setFoldExpanded(This::wxStyledTextCtrl(), Line::integer(), Expanded::bool()) -> ok`

See external documentation.

`getFoldExpanded(This::wxStyledTextCtrl(), Line::integer()) -> bool()`

See external documentation.

`toggleFold(This::wxStyledTextCtrl(), Line::integer()) -> ok`

See external documentation.

`ensureVisible(This::wxStyledTextCtrl(), Line::integer()) -> ok`

See external documentation.

`setFoldFlags(This::wxStyledTextCtrl(), Flags::integer()) -> ok`

See external documentation.

`ensureVisibleEnforcePolicy(This::wxStyledTextCtrl(), Line::integer()) -> ok`

See external documentation.

`setTabIndents(This::wxStyledTextCtrl(), TabIndents::bool()) -> ok`

See external documentation.

`getTabIndents(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setBackSpaceUnIndents(This::wxStyledTextCtrl(), BsUnIndents::bool()) -> ok`

See external documentation.

`getBackSpaceUnIndents(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setMouseDwellTime(This::wxStyledTextCtrl(), PeriodMilliseconds::integer()) -> ok`

See external documentation.

`getMouseDwellTime(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`wordStartPosition(This::wxStyledTextCtrl(), Pos::integer(), OnlyWordCharacters::bool()) -> integer()`

See external documentation.

`wordEndPosition(This::wxStyledTextCtrl(), Pos::integer(), OnlyWordCharacters::bool()) -> integer()`

See external documentation.

`setWrapMode(This::wxStyledTextCtrl(), Mode::integer()) -> ok`

See external documentation.

`getWrapMode(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setWrapVisualFlags(This::wxStyledTextCtrl(), WrapVisualFlags::integer()) -> ok`

See external documentation.

`getWrapVisualFlags(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

**setWrapVisualFlagsLocation**(This::wxStyledTextCtrl(),  
WrapVisualFlagsLocation::integer()) -> ok

See external documentation.

**getWrapVisualFlagsLocation**(This::wxStyledTextCtrl()) -> integer()

See external documentation.

**setWrapStartIndent**(This::wxStyledTextCtrl(), Indent::integer()) -> ok

See external documentation.

**getWrapStartIndent**(This::wxStyledTextCtrl()) -> integer()

See external documentation.

**setLayoutCache**(This::wxStyledTextCtrl(), Mode::integer()) -> ok

See external documentation.

**getLayoutCache**(This::wxStyledTextCtrl()) -> integer()

See external documentation.

**setScrollWidth**(This::wxStyledTextCtrl(), PixelWidth::integer()) -> ok

See external documentation.

**getScrollWidth**(This::wxStyledTextCtrl()) -> integer()

See external documentation.

**textWidth**(This::wxStyledTextCtrl(), Style::integer(), Text::string()) ->  
integer()

See external documentation.

**getEndAtLastLine**(This::wxStyledTextCtrl()) -> bool()

See external documentation.

**textHeight**(This::wxStyledTextCtrl(), Line::integer()) -> integer()

See external documentation.

**setUseVerticalScrollBar**(This::wxStyledTextCtrl(), Show::bool()) -> ok

See external documentation.

**getUseVerticalScrollBar**(This::wxStyledTextCtrl()) -> bool()

See external documentation.

**appendText**(This::wxStyledTextCtrl(), Text::string()) -> ok

See external documentation.

`getTwoPhaseDraw(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setTwoPhaseDraw(This::wxStyledTextCtrl(), TwoPhase::bool()) -> ok`

See external documentation.

`targetFromSelection(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`linesJoin(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`linesSplit(This::wxStyledTextCtrl(), PixelWidth::integer()) -> ok`

See external documentation.

`setFoldMarginColour(This::wxStyledTextCtrl(), UseSetting::bool(),  
Back::colour() (see module wx)) -> ok`

See external documentation.

`setFoldMarginHiColour(This::wxStyledTextCtrl(), UseSetting::bool(),  
Fore::colour() (see module wx)) -> ok`

See external documentation.

`lineDown(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineDownExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineUp(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineUpExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`charLeft(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`charLeftExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`charRight(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`charRightExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordLeft(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordLeftExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordRight(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordRightExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`home(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`homeExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineEnd(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineEndExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`documentStart(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`documentStartExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`documentEnd(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`documentEndExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`pageUp(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`pageUpExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`pageDown(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`pageDownExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`editToggleOvertype(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`cancel(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`deleteBack(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`tab(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`backTab(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`newLine(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`formFeed(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`vCHome(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`vCHomeExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`zoomIn(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`zoomOut(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`delWordLeft(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`delWordRight(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineCut(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineDelete(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineTranspose(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineDuplicate(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lowerCase(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`upperCase(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineScrollDown(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineScrollUp(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`deleteBackNotLine(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`homeDisplay(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`homeDisplayExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineEndDisplay(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineEndDisplayExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`homeWrapExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineEndWrap(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineEndWrapExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`vCHomeWrap(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`vCHomeWrapExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineCopy(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`moveCaretInsideView(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineLength(This::wxStyledTextCtrl(), Line::integer()) -> integer()`

See external documentation.

`braceHighlight(This::wxStyledTextCtrl(), Pos1::integer(), Pos2::integer()) -> ok`

See external documentation.

`braceBadLight(This::wxStyledTextCtrl(), Pos::integer()) -> ok`

See external documentation.

`braceMatch(This::wxStyledTextCtrl(), Pos::integer()) -> integer()`

See external documentation.

`getViewEOL(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setViewEOL(This::wxStyledTextCtrl(), Visible::bool()) -> ok`

See external documentation.



**setModEventMask(This::wxStyledTextCtrl(), Mask::integer()) -> ok**

See external documentation.

**getEdgeColumn(This::wxStyledTextCtrl()) -> integer()**

See external documentation.

**setEdgeColumn(This::wxStyledTextCtrl(), Column::integer()) -> ok**

See external documentation.

**getEdgeMode(This::wxStyledTextCtrl()) -> integer()**

See external documentation.

**getEdgeColour(This::wxStyledTextCtrl()) -> colour() (see module wx)**

See external documentation.

**setEdgeColour(This::wxStyledTextCtrl(), EdgeColour::colour() (see module wx))  
-> ok**

See external documentation.

**searchAnchor(This::wxStyledTextCtrl()) -> ok**

See external documentation.

**searchNext(This::wxStyledTextCtrl(), Flags::integer(), Text::string()) ->  
integer()**

See external documentation.

**searchPrev(This::wxStyledTextCtrl(), Flags::integer(), Text::string()) ->  
integer()**

See external documentation.

**linesOnScreen(This::wxStyledTextCtrl()) -> integer()**

See external documentation.

**usePopUp(This::wxStyledTextCtrl(), AllowPopUp::bool()) -> ok**

See external documentation.

**selectionIsRectangle(This::wxStyledTextCtrl()) -> bool()**

See external documentation.

**setZoom(This::wxStyledTextCtrl(), Zoom::integer()) -> ok**

See external documentation.

`getZoom(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`getModEventMask(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setSTCFocus(This::wxStyledTextCtrl(), Focus::bool()) -> ok`

See external documentation.

`getSTCFocus(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setStatus(This::wxStyledTextCtrl(), StatusCode::integer()) -> ok`

See external documentation.

`getStatus(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setMouseDownCaptures(This::wxStyledTextCtrl(), Captures::bool()) -> ok`

See external documentation.

`getMouseDownCaptures(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setSTCCursor(This::wxStyledTextCtrl(), CursorType::integer()) -> ok`

See external documentation.

`getSTCCursor(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setControlCharSymbol(This::wxStyledTextCtrl(), Symbol::integer()) -> ok`

See external documentation.

`getControlCharSymbol(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`wordPartLeft(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordPartLeftExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordPartRight(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordPartRightExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setVisiblePolicy(This::wxStyledTextCtrl(), VisiblePolicy::integer(),  
VisibleSlop::integer()) -> ok`

See external documentation.

`delLineLeft(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`delLineRight(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`getXOffset(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`chooseCaretX(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setXCaretPolicy(This::wxStyledTextCtrl(), CaretPolicy::integer(),  
CaretSlop::integer()) -> ok`

See external documentation.

`setYCaretPolicy(This::wxStyledTextCtrl(), CaretPolicy::integer(),  
CaretSlop::integer()) -> ok`

See external documentation.

`getPrintWrapMode(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`setHotspotActiveForeground(This::wxStyledTextCtrl(), UseSetting::bool(),  
Fore::colour() (see module wx)) -> ok`

See external documentation.

`setHotspotActiveBackground(This::wxStyledTextCtrl(), UseSetting::bool(),  
Back::colour() (see module wx)) -> ok`

See external documentation.

`setHotspotActiveUnderline(This::wxStyledTextCtrl(), Underline::bool()) -> ok`

See external documentation.

`setHotspotSingleLine(This::wxStyledTextCtrl(), SingleLine::bool()) -> ok`

See external documentation.

`paraDownExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`paraUp(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`paraUpExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`positionBefore(This::wxStyledTextCtrl(), Pos::integer()) -> integer()`

See external documentation.

`positionAfter(This::wxStyledTextCtrl(), Pos::integer()) -> integer()`

See external documentation.

`copyRange(This::wxStyledTextCtrl(), Start::integer(), End::integer()) -> ok`

See external documentation.

`copyText(This::wxStyledTextCtrl(), Length::integer(), Text::string()) -> ok`

See external documentation.

`setSelectionMode(This::wxStyledTextCtrl(), Mode::integer()) -> ok`

See external documentation.

`getSelectionMode(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`lineDownRectExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineUpRectExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`charLeftRectExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`charRightRectExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`homeRectExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`vCHomeRectExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`lineEndRectExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`pageUpRectExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`pageDownRectExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`stutteredPageUp(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`stutteredPageUpExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`stutteredPageDown(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`stutteredPageDownExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordLeftEnd(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordLeftEndExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordRightEnd(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`wordRightEndExtend(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setWhitespaceChars(This::wxStyledTextCtrl(), Characters::string()) -> ok`

See external documentation.

`setCharsDefault(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`autoCompGetCurrent(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`allocate(This::wxStyledTextCtrl(), Bytes::integer()) -> ok`

See external documentation.

`findColumn(This::wxStyledTextCtrl(), Line::integer(), Column::integer()) -> integer()`

See external documentation.

`getCaretSticky(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setCaretSticky(This::wxStyledTextCtrl(), UseCaretStickyBehaviour::bool()) -> ok`

See external documentation.

`toggleCaretSticky(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setPasteConvertEndings(This::wxStyledTextCtrl(), Convert::bool()) -> ok`

See external documentation.

`getPasteConvertEndings(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`selectionDuplicate(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setCaretLineBackAlpha(This::wxStyledTextCtrl(), Alpha::integer()) -> ok`

See external documentation.

`getCaretLineBackAlpha(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`startRecord(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`stopRecord(This::wxStyledTextCtrl()) -> ok`

See external documentation.

`setLexer(This::wxStyledTextCtrl(), Lexer::integer()) -> ok`

See external documentation.

`getLexer(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`colourise(This::wxStyledTextCtrl(), Start::integer(), End::integer()) -> ok`

See external documentation.

`setProperty(This::wxStyledTextCtrl(), Key::string(), Value::string()) -> ok`

See external documentation.

`setKeyWords(This::wxStyledTextCtrl(), KeywordSet::integer(),  
KeyWords::string()) -> ok`

See external documentation.

`setLexerLanguage(This::wxStyledTextCtrl(), Language::string()) -> ok`

See external documentation.

`getProperty(This::wxStyledTextCtrl(), Key::string()) -> string()`

See external documentation.

`getStyleBitsNeeded(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`getCurrentLine(This::wxStyledTextCtrl()) -> integer()`

See external documentation.

`styleSetSpec(This::wxStyledTextCtrl(), StyleNum::integer(), Spec::string()) -  
> ok`

See external documentation.

`styleSetFont(This::wxStyledTextCtrl(), StyleNum::integer(), Font::wxFont()  
(see module wxFont)) -> ok`

See external documentation.

`styleSetFontAttr(This::wxStyledTextCtrl(), StyleNum::integer(),  
Size::integer(), FaceName::string(), Bold::bool(), Italic::bool(),  
Underline::bool()) -> ok`

Equivalent to *styleSetFontAttr(This, StyleNum, Size, FaceName, Bold, Italic, Underline, [])*.

```
styleSetFontAttr(This::wxStyledTextCtrl(), StyleNum::integer(),
Size::integer(), FaceName::string(), Bold::bool(), Italic::bool(),
Underline::bool(), Options::[Option]) -> ok
```

Types:

**Option** = {encoding, WxFontEncoding}

**WxFontEncoding** = integer()

See **external documentation**.

WxFontEncoding is one of ?wxFONTENCODING\_SYSTEM | ?wxFONTENCODING\_DEFAULT | ?wxFONTENCODING\_ISO8859\_1 | ?wxFONTENCODING\_ISO8859\_2 | ?wxFONTENCODING\_ISO8859\_3 | ?wxFONTENCODING\_ISO8859\_4 | ?wxFONTENCODING\_ISO8859\_5 | ?wxFONTENCODING\_ISO8859\_6 | ?wxFONTENCODING\_ISO8859\_7 | ?wxFONTENCODING\_ISO8859\_8 | ?wxFONTENCODING\_ISO8859\_9 | ?wxFONTENCODING\_ISO8859\_10 | ?wxFONTENCODING\_ISO8859\_11 | ?wxFONTENCODING\_ISO8859\_12 | ?wxFONTENCODING\_ISO8859\_13 | ?wxFONTENCODING\_ISO8859\_14 | ?wxFONTENCODING\_ISO8859\_15 | ?wxFONTENCODING\_ISO8859\_MAX | ?wxFONTENCODING\_KOI8 | ?wxFONTENCODING\_KOI8\_U | ?wxFONTENCODING\_ALTERNATIVE | ?wxFONTENCODING\_BULGARIAN | ?wxFONTENCODING\_CP437 | ?wxFONTENCODING\_CP850 | ?wxFONTENCODING\_CP852 | ?wxFONTENCODING\_CP855 | ?wxFONTENCODING\_CP866 | ?wxFONTENCODING\_CP874 | ?wxFONTENCODING\_CP932 | ?wxFONTENCODING\_CP936 | ?wxFONTENCODING\_CP949 | ?wxFONTENCODING\_CP950 | ?wxFONTENCODING\_CP1250 | ?wxFONTENCODING\_CP1251 | ?wxFONTENCODING\_CP1252 | ?wxFONTENCODING\_CP1253 | ?wxFONTENCODING\_CP1254 | ?wxFONTENCODING\_CP1255 | ?wxFONTENCODING\_CP1256 | ?wxFONTENCODING\_CP1257 | ?wxFONTENCODING\_CP12\_MAX | ?wxFONTENCODING\_UTF7 | ?wxFONTENCODING\_UTF8 | ?wxFONTENCODING\_EUC\_JP | ?wxFONTENCODING\_UTF16BE | ?wxFONTENCODING\_UTF16LE | ?wxFONTENCODING\_UTF32BE | ?wxFONTENCODING\_UTF32LE | ?wxFONTENCODING\_MACROMAN | ?wxFONTENCODING\_MACJAPANESE | ?wxFONTENCODING\_MACCHINESETRAD | ?wxFONTENCODING\_MACKOREAN | ?wxFONTENCODING\_MACARABIC | ?wxFONTENCODING\_MACHEBREW | ?wxFONTENCODING\_MACGREEK | ?wxFONTENCODING\_MACCYRILLIC | ?wxFONTENCODING\_MACDEVANAGARI | ?wxFONTENCODING\_MACGURMUKHI | ?wxFONTENCODING\_MACGUJARATI | ?wxFONTENCODING\_MACORIYA | ?wxFONTENCODING\_MACBENGALI | ?wxFONTENCODING\_MACTAMIL | ?wxFONTENCODING\_MACTELUGU | ?wxFONTENCODING\_MACKANNADA | ?wxFONTENCODING\_MACMALAJALAM | ?wxFONTENCODING\_MACSINHALESE | ?wxFONTENCODING\_MACBURMESE | ?wxFONTENCODING\_MACKHMER | ?wxFONTENCODING\_MACTHAI | ?wxFONTENCODING\_MACLAOTIAN | ?wxFONTENCODING\_MACGEORGIAN | ?wxFONTENCODING\_MACARMENIAN | ?wxFONTENCODING\_MACCHINESESIMP | ?wxFONTENCODING\_MACTIBETAN | ?wxFONTENCODING\_MACMONGOLIAN | ?wxFONTENCODING\_MACETHIOPIC | ?wxFONTENCODING\_MACCENTRALEUR | ?wxFONTENCODING\_MACVIATNAMESE | ?wxFONTENCODING\_MACARABICEXT | ?wxFONTENCODING\_MACSYMBOL | ?wxFONTENCODING\_MACDINGBATS | ?wxFONTENCODING\_MACTURKISH | ?wxFONTENCODING\_MACCROATIAN | ?wxFONTENCODING\_MACICELANDIC | ?wxFONTENCODING\_MACROMANIAN | ?wxFONTENCODING\_MACCEL TIC | ?wxFONTENCODING\_MACGAELIC | ?wxFONTENCODING\_MACKEYBOARD | ?wxFONTENCODING\_MAX | ?wxFONTENCODING\_MACMIN | ?wxFONTENCODING\_MACMAX | ?wxFONTENCODING\_UTF16 | ?wxFONTENCODING\_UTF32 | ?wxFONTENCODING\_UNICODE | ?wxFONTENCODING\_GB2312 | ?wxFONTENCODING\_BIG5 | ?wxFONTENCODING\_SHIFT\_JIS



```
styleSetCharacterSet(This::wxStyledTextCtrl(), Style::integer(),
CharacterSet::integer()) -> ok
```

See external documentation.

```
styleSetFontEncoding(This::wxStyledTextCtrl(), Style::integer(),
Encoding::WxFontEncoding) -> ok
```

Types:

**WxFontEncoding = integer()**

See external documentation.

WxFontEncoding is one of ?wxFONTENCODING\_SYSTEM | ?wxFONTENCODING\_DEFAULT | ?wxFONTENCODING\_ISO8859\_1 | ?wxFONTENCODING\_ISO8859\_2 | ?wxFONTENCODING\_ISO8859\_3 | ?wxFONTENCODING\_ISO8859\_4 | ?wxFONTENCODING\_ISO8859\_5 | ?wxFONTENCODING\_ISO8859\_6 | ?wxFONTENCODING\_ISO8859\_7 | ?wxFONTENCODING\_ISO8859\_8 | ?wxFONTENCODING\_ISO8859\_9 | ?wxFONTENCODING\_ISO8859\_10 | ?wxFONTENCODING\_ISO8859\_11 | ?wxFONTENCODING\_ISO8859\_12 | ?wxFONTENCODING\_ISO8859\_13 | ?wxFONTENCODING\_ISO8859\_14 | ?wxFONTENCODING\_ISO8859\_15 | ?wxFONTENCODING\_ISO8859\_MAX | ?wxFONTENCODING\_KOI8 | ?wxFONTENCODING\_KOI8\_U | ?wxFONTENCODING\_ALTERNATIVE | ?wxFONTENCODING\_BULGARIAN | ?wxFONTENCODING\_CP437 | ?wxFONTENCODING\_CP850 | ?wxFONTENCODING\_CP852 | ?wxFONTENCODING\_CP855 | ?wxFONTENCODING\_CP866 | ?wxFONTENCODING\_CP874 | ?wxFONTENCODING\_CP932 | ?wxFONTENCODING\_CP936 | ?wxFONTENCODING\_CP949 | ?wxFONTENCODING\_CP950 | ?wxFONTENCODING\_CP1250 | ?wxFONTENCODING\_CP1251 | ?wxFONTENCODING\_CP1252 | ?wxFONTENCODING\_CP1253 | ?wxFONTENCODING\_CP1254 | ?wxFONTENCODING\_CP1255 | ?wxFONTENCODING\_CP1256 | ?wxFONTENCODING\_CP1257 | ?wxFONTENCODING\_CP12\_MAX | ?wxFONTENCODING\_UTF7 | ?wxFONTENCODING\_UTF8 | ?wxFONTENCODING\_EUC\_JP | ?wxFONTENCODING\_UTF16BE | ?wxFONTENCODING\_UTF16LE | ?wxFONTENCODING\_UTF32BE | ?wxFONTENCODING\_UTF32LE | ?wxFONTENCODING\_MACROMAN | ?wxFONTENCODING\_MACJAPANESE | ?wxFONTENCODING\_MACCHINESETRAD | ?wxFONTENCODING\_MACKOREAN | ?wxFONTENCODING\_MACARABIC | ?wxFONTENCODING\_MACHEBREW | ?wxFONTENCODING\_MACGREEK | ?wxFONTENCODING\_MACCYRILLIC | ?wxFONTENCODING\_MACDEVANAGARI | ?wxFONTENCODING\_MACGURMUKHI | ?wxFONTENCODING\_MACGUJARATI | ?wxFONTENCODING\_MACORIYA | ?wxFONTENCODING\_MACBENGALI | ?wxFONTENCODING\_MACTAMIL | ?wxFONTENCODING\_MACTELUGU | ?wxFONTENCODING\_MACKANNADA | ?wxFONTENCODING\_MACMALAJALAM | ?wxFONTENCODING\_MACSINHALESE | ?wxFONTENCODING\_MACBURMESE | ?wxFONTENCODING\_MACKHMER | ?wxFONTENCODING\_MACTHAI | ?wxFONTENCODING\_MACLAOTIAN | ?wxFONTENCODING\_MACGEORGIAN | ?wxFONTENCODING\_MACARMENIAN | ?wxFONTENCODING\_MACCHINESESIMP | ?wxFONTENCODING\_MACTIBETAN | ?wxFONTENCODING\_MACMONGOLIAN | ?wxFONTENCODING\_MACETHIOPIA | ?wxFONTENCODING\_MACCENTRALEUR | ?wxFONTENCODING\_MACVIATNAMESE | ?wxFONTENCODING\_MACARABICEXT | ?wxFONTENCODING\_MACSYMBOL | ?wxFONTENCODING\_MACDINGBATS | ?wxFONTENCODING\_MACTURKISH | ?wxFONTENCODING\_MACCROATIAN | ?wxFONTENCODING\_MACICELANDIC | ?wxFONTENCODING\_MACROMANIAN | ?wxFONTENCODING\_MACCELTIC | ?wxFONTENCODING\_MACGAELIC | ?wxFONTENCODING\_MACKEYBOARD | ?wxFONTENCODING\_MAX | ?wxFONTENCODING\_MACMIN | ?wxFONTENCODING\_MACMAX | ?wxFONTENCODING\_UTF16 | ?wxFONTENCODING\_UTF32 | ?wxFONTENCODING\_UNICODE | ?wxFONTENCODING\_GB2312 | ?wxFONTENCODING\_BIG5 | ?wxFONTENCODING\_SHIFT\_JIS

`cmdKeyExecute(This::wxStyledTextCtrl(), Cmd::integer()) -> ok`

See external documentation.

`setMargins(This::wxStyledTextCtrl(), Left::integer(), Right::integer()) -> ok`

See external documentation.

`getSelection(This::wxStyledTextCtrl()) -> {StartPos::integer(),  
EndPos::integer() }`

See external documentation.

`pointFromPosition(This::wxStyledTextCtrl(), Pos::integer()) -> {X::integer(),  
Y::integer() }`

See external documentation.

`scrollToLine(This::wxStyledTextCtrl(), Line::integer()) -> ok`

See external documentation.

`scrollToColumn(This::wxStyledTextCtrl(), Column::integer()) -> ok`

See external documentation.

`sendMsg(This::wxStyledTextCtrl(), Msg::integer()) -> integer()`

Equivalent to *sendMsg(This, Msg, [])*.

`sendMsg(This::wxStyledTextCtrl(), Msg::integer(), Options::[Option]) ->  
integer()`

Types:

**Option** = {wp, integer()} | {lp, integer() }

See external documentation.

`setVScrollBar(This::wxStyledTextCtrl(), Bar::wxScrollBar() (see module  
wxScrollBar)) -> ok`

See external documentation.

`setHScrollBar(This::wxStyledTextCtrl(), Bar::wxScrollBar() (see module  
wxScrollBar)) -> ok`

See external documentation.

`getLastKeydownProcessed(This::wxStyledTextCtrl()) -> bool()`

See external documentation.

`setLastKeydownProcessed(This::wxStyledTextCtrl(), Val::bool()) -> ok`

See external documentation.

**saveFile**(This::wxStyledTextCtrl(), Filename::string()) -> bool()

See external documentation.

**loadFile**(This::wxStyledTextCtrl(), Filename::string()) -> bool()

See external documentation.

**doDragOver**(This::wxStyledTextCtrl(), X::integer(), Y::integer(),  
Def::WxDragResult) -> WxDragResult

Types:

WxDragResult = integer()

WxDragResult = integer()

See external documentation.

WxDragResult is one of ?wxDragError | ?wxDragNone | ?wxDragCopy | ?wxDragMove | ?wxDragLink | ?  
wxDragCancel

WxDragResult is one of ?wxDragError | ?wxDragNone | ?wxDragCopy | ?wxDragMove | ?wxDragLink | ?  
wxDragCancel

**doDropText**(This::wxStyledTextCtrl(), X::integer(), Y::integer(),  
Data::string()) -> bool()

See external documentation.

**getUseAntiAliasing**(This::wxStyledTextCtrl()) -> bool()

See external documentation.

**addTextRaw**(This::wxStyledTextCtrl(), Text::binary()) -> ok

See external documentation.

**insertTextRaw**(This::wxStyledTextCtrl(), Pos::integer(), Text::binary()) -> ok

See external documentation.

**getCurLineRaw**(This::wxStyledTextCtrl()) -> {binary(), LinePos::integer()}

See external documentation.

**getLineRaw**(This::wxStyledTextCtrl(), Line::integer()) -> binary()

See external documentation.

**getSelectedTextRaw**(This::wxStyledTextCtrl()) -> binary()

See external documentation.

**getTextRangeRaw**(This::wxStyledTextCtrl(), StartPos::integer(),  
EndPos::integer()) -> binary()

See external documentation.

## **wxStyledTextCtrl**

---

**setTextRaw(This::wxStyledTextCtrl(), Text::binary()) -> ok**

See [external documentation](#).

**getTextRaw(This::wxStyledTextCtrl()) -> binary()**

See [external documentation](#).

**appendTextRaw(This::wxStyledTextCtrl(), Text::binary()) -> ok**

See [external documentation](#).

**destroy(This::wxStyledTextCtrl()) -> ok**

Destroys this object, do not use object again

# wxStyledTextEvent

Erlang module

See external documentation: **wxStyledTextEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*stc\_change, stc\_stylenEEDED, stc\_charadded, stc\_savepointreached, stc\_savepointleft, stc\_romodifyattempt, stc\_key, stc\_doubleclick, stc\_updateui, stc\_modified, stc\_macrorecord, stc\_marginclick, stc\_needsShown, stc\_painted, stc\_userlistselection, stc\_uridropped, stc\_dwellstart, stc\_dwellend, stc\_start\_drag, stc\_drag\_over, stc\_do\_drop, stc\_zoom, stc\_hotspot\_click, stc\_hotspot\_dclick, stc\_calltip\_click, stc\_autocomp\_selection*

See also the message variant `#wxStyledText{}` event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

## DATA TYPES

`wxStyledTextEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

## Exports

`getPosition(This::wxStyledTextEvent()) -> integer()`

See external documentation.

`getKey(This::wxStyledTextEvent()) -> integer()`

See external documentation.

`getModifiers(This::wxStyledTextEvent()) -> integer()`

See external documentation.

`getModificationType(This::wxStyledTextEvent()) -> integer()`

See external documentation.

`getText(This::wxStyledTextEvent()) -> string()`

See external documentation.

`getLength(This::wxStyledTextEvent()) -> integer()`

See external documentation.

`getLinesAdded(This::wxStyledTextEvent()) -> integer()`

See external documentation.

`getLine(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getFoldLevelNow(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getFoldLevelPrev(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getMargin(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getMessage(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getWParam(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getLParam(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getListType(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getX(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getY(This::wxStyledTextEvent()) -> integer()`

See [external documentation](#).

`getDragText(This::wxStyledTextEvent()) -> string()`

See [external documentation](#).

`getDragAllowMove(This::wxStyledTextEvent()) -> bool()`

See [external documentation](#).

`getDragResult(This::wxStyledTextEvent()) -> WxDragResult`

Types:

**WxDragResult** = integer()

See [external documentation](#).

WxDragResult is one of ?wxDragError | ?wxDragNone | ?wxDragCopy | ?wxDragMove | ?wxDragLink | ?wxDragCancel

**getShift(This::wxStyledTextEvent()) -> bool()**

See external documentation.

**getControl(This::wxStyledTextEvent()) -> bool()**

See external documentation.

**getAlt(This::wxStyledTextEvent()) -> bool()**

See external documentation.

## wxSysColourChangedEvent

---

Erlang module

See external documentation: **wxSysColourChangedEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*sys\_colour\_changed*

See also the message variant *#wxSysColourChanged{}* event record type.

This class is derived (and can use functions) from:

*wxEvent*

### DATA TYPES

`wxSysColourChangedEvent ( )`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.



## wxTextAttr

---

Erlang module

See external documentation: **wxTextAttr**.

## DATA TYPES

`wxTextAttr()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxTextAttr()`

See external documentation.

`new(ColText::colour() (see module wx)) -> wxTextAttr()`

Equivalent to `new(ColText, [])`.

`new(ColText::colour() (see module wx), Options::[Option]) -> wxTextAttr()`

Types:

`Option = {colBack, colour() (see module wx)} | {font, wxFont() (see module wxFont)} | {alignment, WxTextAttrAlignment}`

`WxTextAttrAlignment = integer()`

See external documentation.

`WxTextAttrAlignment` is one of `?wxTEXT_ALIGNMENT_DEFAULT` | `?wxTEXT_ALIGNMENT_LEFT` | `?wxTEXT_ALIGNMENT_CENTRE` | `?wxTEXT_ALIGNMENT_CENTER` | `?wxTEXT_ALIGNMENT_RIGHT` | `?wxTEXT_ALIGNMENT_JUSTIFIED`

`getAlignment(This::wxTextAttr()) -> WxTextAttrAlignment`

Types:

`WxTextAttrAlignment = integer()`

See external documentation.

`WxTextAttrAlignment` is one of `?wxTEXT_ALIGNMENT_DEFAULT` | `?wxTEXT_ALIGNMENT_LEFT` | `?wxTEXT_ALIGNMENT_CENTRE` | `?wxTEXT_ALIGNMENT_CENTER` | `?wxTEXT_ALIGNMENT_RIGHT` | `?wxTEXT_ALIGNMENT_JUSTIFIED`

`getBackgroundColour(This::wxTextAttr()) -> colour() (see module wx)`

See external documentation.

`getFont(This::wxTextAttr()) -> wxFont() (see module wxFont)`

See external documentation.

`getLeftIndent(This::wxTextAttr()) -> integer()`

See external documentation.

`getLeftSubIndent(This::wxTextAttr()) -> integer()`

See external documentation.

`getRightIndent(This::wxTextAttr()) -> integer()`

See external documentation.

`getTabs(This::wxTextAttr()) -> [integer()]`

See external documentation.

`getTextColour(This::wxTextAttr()) -> colour() (see module wx)`

See external documentation.

`hasBackgroundColour(This::wxTextAttr()) -> bool()`

See external documentation.

`hasFont(This::wxTextAttr()) -> bool()`

See external documentation.

`hasTextColour(This::wxTextAttr()) -> bool()`

See external documentation.

`getFlags(This::wxTextAttr()) -> integer()`

See external documentation.

`isDefault(This::wxTextAttr()) -> bool()`

See external documentation.

`setAlignment(This::wxTextAttr(), Alignment::WxTextAttrAlignment) -> ok`

Types:

`WxTextAttrAlignment = integer()`

See external documentation.

WxTextAttrAlignment is one of ?wxTEXT\_ALIGNMENT\_DEFAULT | ?wxTEXT\_ALIGNMENT\_LEFT | ?wxTEXT\_ALIGNMENT\_CENTRE | ?wxTEXT\_ALIGNMENT\_CENTER | ?wxTEXT\_ALIGNMENT\_RIGHT | ?wxTEXT\_ALIGNMENT\_JUSTIFIED

`setBackgroundColour(This::wxTextAttr(), ColBack::colour() (see module wx)) -> ok`

See external documentation.

**setFlags(This::wxTextAttr(), Flags::integer()) -> ok**

See external documentation.

**setFont(This::wxTextAttr(), Font::wxFont() (see module wxFont)) -> ok**

Equivalent to *setFont(This, Font, [])*.

**setFont(This::wxTextAttr(), Font::wxFont() (see module wxFont), Options::[Option]) -> ok**

Types:

**Option = {flags, integer()}**

See external documentation.

**setLeftIndent(This::wxTextAttr(), Indent::integer()) -> ok**

Equivalent to *setLeftIndent(This, Indent, [])*.

**setLeftIndent(This::wxTextAttr(), Indent::integer(), Options::[Option]) -> ok**

Types:

**Option = {subIndent, integer()}**

See external documentation.

**setRightIndent(This::wxTextAttr(), Indent::integer()) -> ok**

See external documentation.

**setTabs(This::wxTextAttr(), Tabs::[integer()]) -> ok**

See external documentation.

**setTextColour(This::wxTextAttr(), ColText::colour() (see module wx)) -> ok**

See external documentation.

**destroy(This::wxTextAttr()) -> ok**

Destroys this object, do not use object again

## wxTextCtrl

---

Erlang module

See external documentation: **wxTextCtrl**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxTextCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxTextCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxTextCtrl()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxTextCtrl()`

Types:

**Option** = {value, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

`appendText(This::wxTextCtrl(), Text::string()) -> ok`

See external documentation.

`canCopy(This::wxTextCtrl()) -> bool()`

See external documentation.

`canCut(This::wxTextCtrl()) -> bool()`

See external documentation.

`canPaste(This::wxTextCtrl()) -> bool()`

See external documentation.

`canRedo(This::wxTextCtrl()) -> bool()`

See external documentation.

`canUndo(This::wxTextCtrl()) -> bool()`

See external documentation.

`clear(This::wxTextCtrl()) -> ok`

See external documentation.

`copy(This::wxTextCtrl()) -> ok`

See external documentation.

`create(This::wxTextCtrl(), Parent::wxWindow() (see module wxWindow),  
Id::integer()) -> bool()`

Equivalent to *create(This, Parent, Id, [])*.

`create(This::wxTextCtrl(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Options::[Option]) -> bool()`

Types:

`Option = {value, string()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style,  
integer()} | {validator, wx() (see module wx)}`

See external documentation.

`cut(This::wxTextCtrl()) -> ok`

See external documentation.

`discardEdits(This::wxTextCtrl()) -> ok`

See external documentation.

`emulateKeyPress(This::wxTextCtrl(), Event::wxKeyEvent() (see module  
wxKeyEvent)) -> bool()`

See external documentation.

`getDefaultStyle(This::wxTextCtrl()) -> wxTextAttr() (see module wxTextAttr)`

See external documentation.

`getInsertionPoint(This::wxTextCtrl()) -> integer()`

See external documentation.

`getLastPosition(This::wxTextCtrl()) -> integer()`

See external documentation.

`getLineLength(This::wxTextCtrl(), LineNo::integer()) -> integer()`

See external documentation.

`getLineText(This::wxTextCtrl(), LineNo::integer()) -> string()`

See external documentation.

`getNumberOfLines(This::wxTextCtrl()) -> integer()`

See external documentation.

`getRange(This::wxTextCtrl(), From::integer(), To::integer()) -> string()`

See external documentation.

`getSelection(This::wxTextCtrl()) -> {From::integer(), To::integer()}`

See external documentation.

`getStringSelection(This::wxTextCtrl()) -> string()`

See external documentation.

`getStyle(This::wxTextCtrl(), Position::integer(), Style::wxTextAttr() (see module wxTextAttr)) -> bool()`

See external documentation.

`getValue(This::wxTextCtrl()) -> string()`

See external documentation.

`isEditable(This::wxTextCtrl()) -> bool()`

See external documentation.

`isModified(This::wxTextCtrl()) -> bool()`

See external documentation.

`isMultiLine(This::wxTextCtrl()) -> bool()`

See external documentation.

`isSingleLine(This::wxTextCtrl()) -> bool()`

See external documentation.

`loadFile(This::wxTextCtrl(), File::string()) -> bool()`

Equivalent to `loadFile(This, File, [])`.

`loadFile(This::wxTextCtrl(), File::string(), Options::[Option]) -> bool()`

Types:

`Option = {fileType, integer()}`

See external documentation.

`markDirty(This::wxTextCtrl()) -> ok`

See external documentation.

`paste(This::wxTextCtrl()) -> ok`

See external documentation.

`positionToXY(This::wxTextCtrl(), Pos::integer()) -> {bool(), X::integer(), Y::integer()}`

See external documentation.

`redo(This::wxTextCtrl()) -> ok`

See external documentation.

`remove(This::wxTextCtrl(), From::integer(), To::integer()) -> ok`

See external documentation.

`replace(This::wxTextCtrl(), From::integer(), To::integer(), Value::string()) -> ok`

See external documentation.

`saveFile(This::wxTextCtrl()) -> bool()`

Equivalent to `saveFile(This, [])`.

`saveFile(This::wxTextCtrl(), Options::[Option]) -> bool()`

Types:

`Option = {file, string()} | {fileType, integer()}`

See external documentation.

`setDefaultStyle(This::wxTextCtrl(), Style::wxTextAttr() (see module wxTextAttr)) -> bool()`

See external documentation.

`setEditable(This::wxTextCtrl(), Editable::bool()) -> ok`

See external documentation.

`setInsertionPoint(This::wxTextCtrl(), Pos::integer()) -> ok`

See external documentation.

`setInsertionPointEnd(This::wxTextCtrl()) -> ok`

See external documentation.

`setMaxLength(This::wxTextCtrl(), Len::integer()) -> ok`

See external documentation.

`setSelection(This::wxTextCtrl(), From::integer(), To::integer()) -> ok`

See external documentation.

`setStyle(This::wxTextCtrl(), Start::integer(), End::integer(),  
Style::wxTextAttr() (see module wxTextAttr)) -> bool()`

See external documentation.

`setValue(This::wxTextCtrl(), Value::string()) -> ok`

See external documentation.

`showPosition(This::wxTextCtrl(), Pos::integer()) -> ok`

See external documentation.

`undo(This::wxTextCtrl()) -> ok`

See external documentation.

`writeText(This::wxTextCtrl(), Text::string()) -> ok`

See external documentation.

`xYToPosition(This::wxTextCtrl(), X::integer(), Y::integer()) -> integer()`

See external documentation.

`destroy(This::wxTextCtrl()) -> ok`

Destroys this object, do not use object again



## wxTextDataObject

---

Erlang module

See external documentation: **wxTextDataObject**.

This class is derived (and can use functions) from:  
*wxDataObject*

### DATA TYPES

`wxTextDataObject()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new() -> wxTextDataObject()`

Equivalent to `new([])`.

`new(Options::[Option]) -> wxTextDataObject()`

Types:

**Option** = {text, string()}

See external documentation.

`getTextLength(This::wxTextDataObject()) -> integer()`

See external documentation.

`getText(This::wxTextDataObject()) -> string()`

See external documentation.

`setText(This::wxTextDataObject(), Text::string()) -> ok`

See external documentation.

`destroy(This::wxTextDataObject()) -> ok`

Destroys this object, do not use object again

## wxTextEntryDialog

---

Erlang module

See external documentation: **wxTextEntryDialog**.

This class is derived (and can use functions) from:

*wxDialog*

*wxTopLevelWindow*

*wxWindow*

*wxEvtHandler*

### DATA TYPES

`wxTextEntryDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new(Parent::wxWindow() (see module wxWindow), Message::string()) -> wxTextEntryDialog()`

Equivalent to `new(Parent, Message, [])`.

`new(Parent::wxWindow() (see module wxWindow), Message::string(), Options::[Option]) -> wxTextEntryDialog()`

Types:

**Option** = {caption, string()} | {value, string()} | {style, integer()} | {pos, {X::integer(), Y::integer()}}

See external documentation.

`getValue(This::wxTextEntryDialog()) -> string()`

See external documentation.

`setValue(This::wxTextEntryDialog(), Val::string()) -> ok`

See external documentation.

`destroy(This::wxTextEntryDialog()) -> ok`

Destroys this object, do not use object again

# wxToggleButton

---

Erlang module

See external documentation: **wxToggleButton**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxToggleButton()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxToggleButton()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string())  
-> wxToggleButton()`

Equivalent to `new(Parent, Id, Label, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Label::string(),  
Options::[Option]) -> wxToggleButton()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} |  
{validator, wx() (see module wx)}

See external documentation.

`create(This::wxToggleButton(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string()) -> bool()`

Equivalent to `create(This, Parent, Id, Label, [])`.

`create(This::wxToggleButton(), Parent::wxWindow() (see module wxWindow),  
Id::integer(), Label::string(), Options::[Option]) -> bool()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} |  
{validator, wx() (see module wx)}

See external documentation.

`getValue(This::wxToggleButton()) -> bool()`

See external documentation.

## **wxToggleButton**

---

**setValue(This::wxToggleButton(), State::bool()) -> ok**

See **external documentation**.

**destroy(This::wxToggleButton()) -> ok**

Destroys this object, do not use object again

# wxToolBar

Erlang module

See external documentation: **wxToolBar**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxToolBar()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`addControl(This::wxToolBar(), Control::wxControl() (see module wxControl)) -> wx() (see module wx)`

See external documentation.

`addSeparator(This::wxToolBar()) -> wx() (see module wx)`

See external documentation.

`addTool(This::wxToolBar(), Tool::wx() (see module wx)) -> wx() (see module wx)`

See external documentation.

`addTool(This::wxToolBar(), Toolid::integer(), Bitmap::wxBitmap() (see module wxBitmap)) -> wx() (see module wx)`

Equivalent to `addTool(This, Toolid, Bitmap, [])`.

`addTool(This::wxToolBar(), Toolid::integer(), X::string() | term(), X::term()) -> wx() (see module wx)`

See external documentation.

Alternatives:

`addTool(This::wxToolBar(), Toolid::integer(), Label::string(), Bitmap::wxBitmap:wxBitmap()) -> addTool(This, Toolid, Label, Bitmap, [])`

`addTool(This::wxToolBar(), Toolid::integer(), Bitmap::wxBitmap:wxBitmap(), BmpDisabled::wxBitmap:wxBitmap()) -> addTool(This, Toolid, Bitmap, BmpDisabled, [])`

`addTool(This::wxToolBar(), Toolid::integer(), Bitmap::wxBitmap:wxBitmap(), [Option]) -> wx:wx()`

Option = {shortHelpString, string()} | {longHelpString, string()}

```
addTool(This::wxToolBar(), Toolid::integer(), X::string() | term(),  
X::term(), X::term()) -> wx() (see module wx)
```

See **external documentation**.

Alternatives:

```
addTool(This::wxToolBar(), Toolid::integer(), Label::string(),  
Bitmap::wxBitmap:wxBitmap(), BmpDisabled::wxBitmap:wxBitmap()) ->  
addTool(This, Toolid, Label, Bitmap, BmpDisabled, [])
```

```
addTool(This::wxToolBar(), Toolid::integer(), Label::string(),  
Bitmap::wxBitmap:wxBitmap(), [Option]) -> wx:wx()  
Option = {shortHelp, string()} | {kind, WxItemKind}  
WxItemKind = integer()  
WxItemKind is one of ?wxITEM_SEPARATOR | ?wxITEM_NORMAL | ?wxITEM_CHECK | ?wxITEM_RADIO  
| ?wxITEM_MAX
```

```
addTool(This::wxToolBar(), Toolid::integer(), Bitmap::wxBitmap:wxBitmap(),  
BmpDisabled::wxBitmap:wxBitmap(), [Option]) -> wx:wx()  
Option = {toggle, bool()} | {clientData, wx:wx()} | {shortHelpString, string()} | {longHelpString, string()}
```

```
addTool(This::wxToolBar(), Toolid::integer(), X::term() | string(),  
X::term(), X::bool() | term(), X::integer() | term()) -> wx() (see module wx)
```

See **external documentation**.

Alternatives:

```
addTool(This::wxToolBar(), Toolid::integer(), Bitmap::wxBitmap:wxBitmap(),  
BmpDisabled::wxBitmap:wxBitmap(), Toggle::bool(), XPos::integer()) ->  
addTool(This, Toolid, Bitmap, BmpDisabled, Toggle, XPos, [])
```

```
addTool(This::wxToolBar(), Toolid::integer(), Label::string(),  
Bitmap::wxBitmap:wxBitmap(), BmpDisabled::wxBitmap:wxBitmap(), [Option]) ->  
wx:wx()  
Option = {kind, WxItemKind} | {shortHelp, string()} | {longHelp, string()} | {data, wx:wx()}  
WxItemKind = integer()  
WxItemKind is one of ?wxITEM_SEPARATOR | ?wxITEM_NORMAL | ?wxITEM_CHECK | ?wxITEM_RADIO  
| ?wxITEM_MAX
```

```
addTool(This::wxToolBar(), Toolid::integer(), Bitmap::wxBitmap() (see module  
wxBitmap), BmpDisabled::wxBitmap() (see module wxBitmap), Toggle::bool(),  
XPos::integer(), Options::[Option]) -> wx() (see module wx)
```

Types:

**Option = {yPos, integer()} | {clientData, wx() (see module wx)} | {shortHelp, string()} | {longHelp, string()}**

See **external documentation**.

```
addCheckTool(This::wxToolBar(), Toolid::integer(), Label::string(),  
Bitmap::wxBitmap() (see module wxBitmap)) -> wx() (see module wx)
```

Equivalent to *addCheckTool(This, Toolid, Label, Bitmap, [])*.

`addCheckTool(This::wxToolBar(), Toolid::integer(), Label::string(),  
Bitmap::wxBitmap() (see module wxBitmap), Options::[Option]) -> wx() (see  
module wx)`

Types:

`Option = {bmpDisabled, wxBitmap() (see module wxBitmap)} | {shortHelp, string()} | {longHelp, string()}  
| {data, wx() (see module wx)}`

See external documentation.

`addRadioTool(This::wxToolBar(), Toolid::integer(), Label::string(),  
Bitmap::wxBitmap() (see module wxBitmap)) -> wx() (see module wx)`

Equivalent to `addRadioTool(This, Toolid, Label, Bitmap, [])`.

`addRadioTool(This::wxToolBar(), Toolid::integer(), Label::string(),  
Bitmap::wxBitmap() (see module wxBitmap), Options::[Option]) -> wx() (see  
module wx)`

Types:

`Option = {bmpDisabled, wxBitmap() (see module wxBitmap)} | {shortHelp, string()} | {longHelp, string()}  
| {data, wx() (see module wx)}`

See external documentation.

`deleteTool(This::wxToolBar(), Toolid::integer()) -> bool()`

See external documentation.

`deleteToolByPos(This::wxToolBar(), Pos::integer()) -> bool()`

See external documentation.

`enableTool(This::wxToolBar(), Toolid::integer(), Enable::bool()) -> ok`

See external documentation.

`findById(This::wxToolBar(), Toolid::integer()) -> wx() (see module wx)`

See external documentation.

`findControl(This::wxToolBar(), Toolid::integer()) -> wxControl() (see module  
wxControl)`

See external documentation.

`findToolForPosition(This::wxToolBar(), X::integer(), Y::integer()) -> wx()  
(see module wx)`

See external documentation.

`getToolSize(This::wxToolBar()) -> {W::integer(), H::integer()}`

See external documentation.

`getToolBitmapSize(This::wxToolBar()) -> {W::integer(), H::integer()}`

See external documentation.

`getMargins(This::wxToolBar()) -> {W::integer(), H::integer()}`

See external documentation.

`getToolEnabled(This::wxToolBar(), Toolid::integer()) -> bool()`

See external documentation.

`getToolLongHelp(This::wxToolBar(), Toolid::integer()) -> string()`

See external documentation.

`getToolPacking(This::wxToolBar()) -> integer()`

See external documentation.

`getToolPos(This::wxToolBar(), Id::integer()) -> integer()`

See external documentation.

`getToolSeparation(This::wxToolBar()) -> integer()`

See external documentation.

`getToolShortHelp(This::wxToolBar(), Toolid::integer()) -> string()`

See external documentation.

`getToolState(This::wxToolBar(), Toolid::integer()) -> bool()`

See external documentation.

`insertControl(This::wxToolBar(), Pos::integer(), Control::wxControl() (see module wxControl)) -> wx() (see module wx)`

See external documentation.

`insertSeparator(This::wxToolBar(), Pos::integer()) -> wx() (see module wx)`

See external documentation.

`insertTool(This::wxToolBar(), Pos::integer(), Tool::wx() (see module wx)) -> wx() (see module wx)`

See external documentation.

`insertTool(This::wxToolBar(), Pos::integer(), Toolid::integer(), Bitmap::wxBitmap() (see module wxBitmap)) -> wx() (see module wx)`

Equivalent to `insertTool(This, Pos, Toolid, Bitmap, [])`.



```
insertTool(This::wxToolBar(), Pos::integer(), Toolid::integer(), X::string()
| term(), X::term()) -> wx() (see module wx)
```

See external documentation.

Alternatives:

```
insertTool(This::wxToolBar(),      Pos::integer(),      Toolid::integer(),
Label::string(),      Bitmap::wxBitmap:wxBitmap())      ->
insertTool(This,Pos,Toolid,Label,Bitmap, [])
```

```
insertTool(This::wxToolBar(),      Pos::integer(),      Toolid::integer(),
Bitmap::wxBitmap:wxBitmap(), [Option]) -> wx:wx()
Option = {bmpDisabled, wxBitmap:wxBitmap()} | {toggle, bool()} | {clientData, wx:wx()} | {shortHelp, string()} |
{longHelp, string()}
```

```
insertTool(This::wxToolBar(), Pos::integer(), Toolid::integer(),
Label::string(), Bitmap::wxBitmap() (see module wxBitmap), Options::[Option])
-> wx() (see module wx)
```

Types:

```
Option = {bmpDisabled, wxBitmap() (see module wxBitmap)} | {kind, WxItemKind} | {shortHelp,
string()} | {longHelp, string()} | {clientData, wx() (see module wx)}
WxItemKind = integer()
```

See external documentation.

WxItemKind is one of ?wxITEM\_SEPARATOR | ?wxITEM\_NORMAL | ?wxITEM\_CHECK | ?wxITEM\_RADIO  
| ?wxITEM\_MAX

```
realize(This::wxToolBar()) -> bool()
```

See external documentation.

```
removeTool(This::wxToolBar(), Toolid::integer()) -> wx() (see module wx)
```

See external documentation.

```
setMargins(This::wxToolBar(), X::integer(), Y::integer()) -> ok
```

See external documentation.

```
setToolBitmapSize(This::wxToolBar(), Size::{W::integer(), H::integer()}) ->
ok
```

See external documentation.

```
setToolLongHelp(This::wxToolBar(), Toolid::integer(), HelpString::string()) -
> ok
```

See external documentation.

```
setToolPacking(This::wxToolBar(), Packing::integer()) -> ok
```

See external documentation.

```
setToolShortHelp(This::wxToolBar(), Id::integer(), HelpString::string()) ->  
ok
```

See [external documentation](#).

```
setToolSeparation(This::wxToolBar(), Separation::integer()) -> ok
```

See [external documentation](#).

```
toggleTool(This::wxToolBar(), Toolid::integer(), Toggle::bool()) -> ok
```

See [external documentation](#).

## wxToolTip

---

Erlang module

See external documentation: **wxToolTip**.

### DATA TYPES

`wxToolTip()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`enable(Flag::bool()) -> ok`**

See external documentation.

**`setDelay(Msecs::integer()) -> ok`**

See external documentation.

**`new(Tip::string()) -> wxToolTip()`**

See external documentation.

**`setTip(This::wxToolTip(), Tip::string()) -> ok`**

See external documentation.

**`getTip(This::wxToolTip()) -> string()`**

See external documentation.

**`getWindow(This::wxToolTip()) -> wxWindow()`** (see module `wxWindow`)

See external documentation.

**`destroy(This::wxToolTip()) -> ok`**

Destroys this object, do not use object again

## wxToolbook

---

Erlang module

See external documentation: **wxToolbook**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxToolbook()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxToolbook()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxToolbook()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxToolbook()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`addPage(This::wxToolbook(), Page::wxWindow() (see module wxWindow), Text::string()) -> bool()`

Equivalent to `addPage(This, Page, Text, [])`.

`addPage(This::wxToolbook(), Page::wxWindow() (see module wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

**Option** = {bSelect, bool()} | {imageId, integer()}

See external documentation.

`advanceSelection(This::wxToolbook()) -> ok`

Equivalent to `advanceSelection(This, [])`.

`advanceSelection(This::wxToolbook(), Options::[Option]) -> ok`

Types:

**Option** = {forward, bool()}

See external documentation.

**assignImageList**(This::wxToolbook(), ImageList::wxImageList() (see module wxImageList)) -> ok

See external documentation.

**create**(This::wxToolbook(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()

Equivalent to *create(This, Parent, Id, [])*.

**create**(This::wxToolbook(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

**deleteAllPages**(This::wxToolbook()) -> bool()

See external documentation.

**deletePage**(This::wxToolbook(), N::integer()) -> bool()

See external documentation.

**removePage**(This::wxToolbook(), N::integer()) -> bool()

See external documentation.

**getCurrentPage**(This::wxToolbook()) -> wxWindow() (see module wxWindow)

See external documentation.

**getImageList**(This::wxToolbook()) -> wxImageList() (see module wxImageList)

See external documentation.

**getPage**(This::wxToolbook(), N::integer()) -> wxWindow() (see module wxWindow)

See external documentation.

**getPageCount**(This::wxToolbook()) -> integer()

See external documentation.

**getPageImage**(This::wxToolbook(), N::integer()) -> integer()

See external documentation.

**getPageText**(This::wxToolbook(), N::integer()) -> string()

See external documentation.

`getSelection(This::wxToolbook()) -> integer()`

See external documentation.

`hitTest(This::wxToolbook(), Pt::{X::integer(), Y::integer()}) -> {integer(),  
Flags::integer()}`

See external documentation.

`insertPage(This::wxToolbook(), N::integer(), Page::wxWindow() (see module  
wxWindow), Text::string()) -> bool()`

Equivalent to `insertPage(This, N, Page, Text, [])`.

`insertPage(This::wxToolbook(), N::integer(), Page::wxWindow() (see module  
wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

`Option = {bSelect, bool()} | {imageId, integer()}`

See external documentation.

`setImageList(This::wxToolbook(), ImageList::wxImageList() (see module  
wxImageList)) -> ok`

See external documentation.

`setPageSize(This::wxToolbook(), Size::{W::integer(), H::integer()}) -> ok`

See external documentation.

`setPageImage(This::wxToolbook(), N::integer(), ImageId::integer()) -> bool()`

See external documentation.

`setPageText(This::wxToolbook(), N::integer(), StrText::string()) -> bool()`

See external documentation.

`setSelection(This::wxToolbook(), N::integer()) -> integer()`

See external documentation.

`changeSelection(This::wxToolbook(), N::integer()) -> integer()`

See external documentation.

`destroy(This::wxToolbook()) -> ok`

Destroys this object, do not use object again

# wxTopLevelWindow

---

Erlang module

See external documentation: **wxTopLevelWindow**.

This class is derived (and can use functions) from:

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxTopLevelWindow()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`getIcon(This::wxTopLevelWindow()) -> wxIcon()` (see module `wxIcon`)

See external documentation.

`getIcons(This::wxTopLevelWindow()) -> wxIconBundle()` (see module `wxIconBundle`)

See external documentation.

`getTitle(This::wxTopLevelWindow()) -> string()`

See external documentation.

`isActive(This::wxTopLevelWindow()) -> bool()`

See external documentation.

`iconize(This::wxTopLevelWindow()) -> ok`

Equivalent to `iconize(This, [])`.

`iconize(This::wxTopLevelWindow(), Options::[Option]) -> ok`

Types:

**Option** = {`iconize`, `bool()`}

See external documentation.

`isFullScreen(This::wxTopLevelWindow()) -> bool()`

See external documentation.

`isIconized(This::wxTopLevelWindow()) -> bool()`

See external documentation.

`isMaximized(This::wxTopLevelWindow()) -> bool()`

See [external documentation](#).

`maximize(This::wxTopLevelWindow()) -> ok`

Equivalent to `maximize(This, [])`.

`maximize(This::wxTopLevelWindow(), Options::[Option]) -> ok`

Types:

**Option** = {maximize, bool()}

See [external documentation](#).

`requestUserAttention(This::wxTopLevelWindow()) -> ok`

Equivalent to `requestUserAttention(This, [])`.

`requestUserAttention(This::wxTopLevelWindow(), Options::[Option]) -> ok`

Types:

**Option** = {flags, integer()}

See [external documentation](#).

`setIcon(This::wxTopLevelWindow(), Icon::wxIcon() (see module wxIcon)) -> ok`

See [external documentation](#).

`setIcons(This::wxTopLevelWindow(), Icons::wxIconBundle() (see module wxIconBundle)) -> ok`

See [external documentation](#).

`centerOnScreen(This::wxTopLevelWindow()) -> ok`

Equivalent to `centerOnScreen(This, [])`.

`centerOnScreen(This::wxTopLevelWindow(), Options::[Option]) -> ok`

Types:

**Option** = {dir, integer()}

See [external documentation](#).

`centreOnScreen(This::wxTopLevelWindow()) -> ok`

Equivalent to `centreOnScreen(This, [])`.

`centreOnScreen(This::wxTopLevelWindow(), Options::[Option]) -> ok`

Types:

**Option** = {dir, integer()}

See [external documentation](#).



```
setShape(This::wxTopLevelWindow(), Region::wxRegion() (see module wxRegion))  
-> bool()
```

See external documentation.

```
setTitle(This::wxTopLevelWindow(), Title::string()) -> ok
```

See external documentation.

```
showFullScreen(This::wxTopLevelWindow(), Show::bool()) -> bool()
```

Equivalent to *showFullScreen(This, Show, [])*.

```
showFullScreen(This::wxTopLevelWindow(), Show::bool(), Options::[Option]) ->  
bool()
```

Types:

**Option** = {style, integer()}

See external documentation.

## wxTreeCtrl

---

Erlang module

See external documentation: **wxTreeCtrl**.

Note: The representation of `treeItemId()` have changed from the original class implementation to be an semi-opaque type, Equality between `TreeItemId`'s can be tested and zero means that the `TreeItem` is invalid.

## DATA TYPES

`wxTreeCtrl()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxTreeCtrl()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow)) -> wxTreeCtrl()`

Equivalent to `new(Parent, [])`.

`new(Parent::wxWindow() (see module wxWindow), Options::[Option]) -> wxTreeCtrl()`

Types:

**Option** = {`id`, `integer()`} | {`pos`, {`X`::`integer()`, `Y`::`integer()`} } | {`size`, {`W`::`integer()`, `H`::`integer()`} } | {`style`, `integer()`} | {`validator`, `wx()` (see module `wx`)}

See external documentation.

`addRoot(This::wxTreeCtrl(), Text::string()) -> integer()`

Equivalent to `addRoot(This, Text, [])`.

`addRoot(This::wxTreeCtrl(), Text::string(), Options::[Option]) -> integer()`

Types:

**Option** = {`image`, `integer()`} | {`selectedImage`, `integer()`} | {`data`, `term()`}

See external documentation.

`appendItem(This::wxTreeCtrl(), Parent::integer(), Text::string()) -> integer()`

Equivalent to `appendItem(This, Parent, Text, [])`.

`appendItem(This::wxTreeCtrl(), Parent::integer(), Text::string(), Options::[Option]) -> integer()`

Types:

**Option** = {`image`, `integer()`} | {`selectedImage`, `integer()`} | {`data`, `term()`}

See external documentation.

```
assignImageList(This::wxTreeCtrl(), ImageList::wxImageList() (see module wxImageList)) -> ok
```

See external documentation.

```
assignStateImageList(This::wxTreeCtrl(), ImageList::wxImageList() (see module wxImageList)) -> ok
```

See external documentation.

```
collapse(This::wxTreeCtrl(), Item::integer()) -> ok
```

See external documentation.

```
collapseAndReset(This::wxTreeCtrl(), Item::integer()) -> ok
```

See external documentation.

```
create(This::wxTreeCtrl(), Parent::wxWindow() (see module wxWindow)) -> bool()
```

Equivalent to *create(This, Parent, [])*.

```
create(This::wxTreeCtrl(), Parent::wxWindow() (see module wxWindow), Options::[Option]) -> bool()
```

Types:

**Option** = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx() (see module wx)}

See external documentation.

```
delete(This::wxTreeCtrl(), Item::integer()) -> ok
```

See external documentation.

```
deleteAllItems(This::wxTreeCtrl()) -> ok
```

See external documentation.

```
deleteChildren(This::wxTreeCtrl(), Item::integer()) -> ok
```

See external documentation.

```
editLabel(This::wxTreeCtrl(), Item::integer()) -> wxTextCtrl() (see module wxTextCtrl)
```

See external documentation.

```
ensureVisible(This::wxTreeCtrl(), Item::integer()) -> ok
```

See external documentation.

`expand(This::wxTreeCtrl(), Item::integer()) -> ok`

See [external documentation](#).

`getBoundingRect(This::wxTreeCtrl(), Item::integer(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> bool()`

Equivalent to `getBoundingRect(This, Item, Rect, [])`.

`getBoundingRect(This::wxTreeCtrl(), Item::integer(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}, Options::[Option]) -> bool()`

Types:

`Option = {textOnly, bool()}`

See [external documentation](#).

`getChildrenCount(This::wxTreeCtrl(), Item::integer()) -> integer()`

Equivalent to `getChildrenCount(This, Item, [])`.

`getChildrenCount(This::wxTreeCtrl(), Item::integer(), Options::[Option]) -> integer()`

Types:

`Option = {recursively, bool()}`

See [external documentation](#).

`getCount(This::wxTreeCtrl()) -> integer()`

See [external documentation](#).

`getEditControl(This::wxTreeCtrl()) -> wxTextCtrl() (see module wxTextCtrl)`

See [external documentation](#).

`getFirstChild(This::wxTreeCtrl(), Item::integer()) -> {integer(), Cookie::integer()}`

See [external documentation](#).

`getNextChild(This::wxTreeCtrl(), Item::integer(), Cookie::integer()) -> {integer(), Cookie::integer()}`

See [external documentation](#).

`getFirstVisibleItem(This::wxTreeCtrl()) -> integer()`

See [external documentation](#).

`getImageList(This::wxTreeCtrl()) -> wxImageList() (see module wxImageList)`

See [external documentation](#).

`getIndent(This::wxTreeCtrl()) -> integer()`

See external documentation.

`getItemBackgroundColour(This::wxTreeCtrl(), Item::integer()) -> colour()` (see module `wx`)

See external documentation.

`getItemData(This::wxTreeCtrl(), Item::integer()) -> term()`

See external documentation.

`getItemFont(This::wxTreeCtrl(), Item::integer()) -> wxFont()` (see module `wxFont`)

See external documentation.

`getItemImage(This::wxTreeCtrl(), Item::integer()) -> integer()`

See external documentation.

`getItemImage(This::wxTreeCtrl(), Item::integer(), Options::[Option]) -> integer()`

Types:

`Option = {which, WxTreeItemIcon}`

`WxTreeItemIcon = integer()`

See external documentation.

`WxTreeItemIcon` is one of `?wxTreeItemIcon_Normal` | `?wxTreeItemIcon_Selected` | `?wxTreeItemIcon_Expanded` | `?wxTreeItemIcon_SelectedExpanded` | `?wxTreeItemIcon_Max`

`getItemText(This::wxTreeCtrl(), Item::integer()) -> string()`

See external documentation.

`getItemTextColour(This::wxTreeCtrl(), Item::integer()) -> colour()` (see module `wx`)

See external documentation.

`getLastChild(This::wxTreeCtrl(), Item::integer()) -> integer()`

See external documentation.

`getNextSibling(This::wxTreeCtrl(), Item::integer()) -> integer()`

See external documentation.

`getNextVisible(This::wxTreeCtrl(), Item::integer()) -> integer()`

See external documentation.

`getItemParent(This::wxTreeCtrl(), Item::integer()) -> integer()`

See external documentation.

`getPrevSibling(This::wxTreeCtrl(), Item::integer()) -> integer()`

See external documentation.

`getPrevVisible(This::wxTreeCtrl(), Item::integer()) -> integer()`

See external documentation.

`getRootItem(This::wxTreeCtrl()) -> integer()`

See external documentation.

`getSelection(This::wxTreeCtrl()) -> integer()`

See external documentation.

`getSelections(This::wxTreeCtrl()) -> {integer(), Val::[integer()]}`

See external documentation.

`getStateImageList(This::wxTreeCtrl()) -> wxImageList() (see module wxImageList)`

See external documentation.

`hitTest(This::wxTreeCtrl(), Point::{X::integer(), Y::integer()}) -> integer()`

See external documentation.

`insertItem(This::wxTreeCtrl(), Parent::integer(), Pos::integer(), Text::string()) -> integer()`

Equivalent to *insertItem(This, Parent, Pos, Text, [])*.

`insertItem(This::wxTreeCtrl(), Parent::integer(), Pos::integer(), Text::string(), Options::[Option]) -> integer()`

Types:

**Option** = {image, integer()} | {setImage, integer()} | {data, term()}

See external documentation.

`isBold(This::wxTreeCtrl(), Item::integer()) -> bool()`

See external documentation.

`isExpanded(This::wxTreeCtrl(), Item::integer()) -> bool()`

See external documentation.

`isSelected(This::wxTreeCtrl(), Item::integer()) -> bool()`

See external documentation.

`isVisible(This::wxTreeCtrl(), Item::integer()) -> bool()`

See external documentation.

```
itemHasChildren(This::wxTreeCtrl(), Item::integer()) -> bool()
```

See external documentation.

```
prependItem(This::wxTreeCtrl(), Parent::integer(), Text::string()) -> integer()
```

Equivalent to *prependItem(This, Parent, Text, [])*.

```
prependItem(This::wxTreeCtrl(), Parent::integer(), Text::string(), Options::[Option]) -> integer()
```

Types:

**Option** = {image, integer()} | {selectedImage, integer()} | {data, term() }

See external documentation.

```
scrollTo(This::wxTreeCtrl(), Item::integer()) -> ok
```

See external documentation.

```
selectItem(This::wxTreeCtrl(), Item::integer()) -> ok
```

See external documentation.

```
selectItem(This::wxTreeCtrl(), Item::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {select, bool() }

See external documentation.

```
setIndent(This::wxTreeCtrl(), Indent::integer()) -> ok
```

See external documentation.

```
setImageList(This::wxTreeCtrl(), ImageList::wxImageList() (see module wxImageList)) -> ok
```

See external documentation.

```
setItemBackgroundColour(This::wxTreeCtrl(), Item::integer(), Col::colour() (see module wx)) -> ok
```

See external documentation.

```
setItemBold(This::wxTreeCtrl(), Item::integer()) -> ok
```

Equivalent to *setItemBold(This, Item, [])*.

```
setItemBold(This::wxTreeCtrl(), Item::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {bold, bool() }

See external documentation.

```
setItemData(This::wxTreeCtrl(), Item::integer(), Data::term()) -> ok
```

See [external documentation](#).

```
setItemDropHighlight(This::wxTreeCtrl(), Item::integer()) -> ok
```

Equivalent to *setItemDropHighlight(This, Item, [])*.

```
setItemDropHighlight(This::wxTreeCtrl(), Item::integer(), Options::[Option])  
-> ok
```

Types:

**Option** = {highlight, bool()}

See [external documentation](#).

```
setItemFont(This::wxTreeCtrl(), Item::integer(), Font::wxFont() (see module  
wxFont)) -> ok
```

See [external documentation](#).

```
setItemHasChildren(This::wxTreeCtrl(), Item::integer()) -> ok
```

Equivalent to *setItemHasChildren(This, Item, [])*.

```
setItemHasChildren(This::wxTreeCtrl(), Item::integer(), Options::[Option]) ->  
ok
```

Types:

**Option** = {has, bool()}

See [external documentation](#).

```
setItemImage(This::wxTreeCtrl(), Item::integer(), Image::integer()) -> ok
```

See [external documentation](#).

```
setItemImage(This::wxTreeCtrl(), Item::integer(), Image::integer(), Options::  
[Option]) -> ok
```

Types:

**Option** = {which, WxTreeItemIcon}

**WxTreeItemIcon** = integer()

See [external documentation](#).

WxTreeItemIcon is one of ?wxTreeItemIcon\_Normal | ?wxTreeItemIcon\_Selected | ?wxTreeItemIcon\_Expanded | ?wxTreeItemIcon\_SelectedExpanded | ?wxTreeItemIcon\_Max

```
setItemText(This::wxTreeCtrl(), Item::integer(), Text::string()) -> ok
```

See [external documentation](#).

```
setItemTextColour(This::wxTreeCtrl(), Item::integer(), Col::colour() (see  
module wx)) -> ok
```

See [external documentation](#).



`setStateImageList(This::wxTreeCtrl(), ImageList::wxImageList() (see module wxImageList)) -> ok`

See external documentation.

`setWindowStyle(This::wxTreeCtrl(), Styles::integer()) -> ok`

See external documentation.

`sortChildren(This::wxTreeCtrl(), Item::integer()) -> ok`

See external documentation.

`toggle(This::wxTreeCtrl(), Item::integer()) -> ok`

See external documentation.

`toggleItemSelection(This::wxTreeCtrl(), Item::integer()) -> ok`

See external documentation.

`unselect(This::wxTreeCtrl()) -> ok`

See external documentation.

`unselectAll(This::wxTreeCtrl()) -> ok`

See external documentation.

`unselectItem(This::wxTreeCtrl(), Item::integer()) -> ok`

See external documentation.

`destroy(This::wxTreeCtrl()) -> ok`

Destroys this object, do not use object again

## wxTreeEvent

---

Erlang module

See external documentation: **wxTreeEvent**.

Use *wxEvtHandler:connect/3* with EventType:

<i>command_tree_begin_drag,</i>	<i>command_tree_begin_rdrag,</i>	<i>command_tree_begin_label_edit,</i>	
<i>command_tree_end_label_edit,</i>	<i>command_tree_delete_item,</i>	<i>command_tree_get_info,</i>	<i>command_tree_set_info,</i>
<i>command_tree_item_expanded,</i>	<i>command_tree_item_expanding,</i>	<i>command_tree_item_collapsed,</i>	
<i>command_tree_item_collapsing,</i>	<i>command_tree_sel_changed,</i>	<i>command_tree_sel_changing,</i>	
<i>command_tree_key_down,</i>	<i>command_tree_item_activated,</i>	<i>command_tree_item_right_click,</i>	
<i>command_tree_item_middle_click,</i>	<i>command_tree_end_drag,</i>	<i>command_tree_state_image_click,</i>	
<i>command_tree_item_gettooltip,</i>	<i>command_tree_item_menu</i>		

See also the message variant *#wxTree{}* event record type.

This class is derived (and can use functions) from:

*wxNotifyEvent*

*wxCommandEvent*

*wxEvent*

## DATA TYPES

*wxTreeEvent()*

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

**getKeyCode(This::wxTreeEvent()) -> integer()**

See external documentation.

**getItem(This::wxTreeEvent()) -> integer()**

See external documentation.

**getKeyEvent(This::wxTreeEvent()) -> wxKeyEvent()** (see module *wxKeyEvent*)

See external documentation.

**getLabel(This::wxTreeEvent()) -> string()**

See external documentation.

**getOldItem(This::wxTreeEvent()) -> integer()**

See external documentation.

**getPoint(This::wxTreeEvent()) -> {X::integer(), Y::integer()}**

See external documentation.

**isEditCancelled(This::wxTreeEvent()) -> bool()**

See external documentation.

**setToolTip(This::wxTreeEvent(), ToolTip::string()) -> ok**

See external documentation.

## wxTreebook

---

Erlang module

See external documentation: **wxTreebook**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

## DATA TYPES

`wxTreebook()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxTreebook()`

See external documentation.

`new(Parent::wxWindow() (see module wxWindow), Id::integer()) -> wxTreebook()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> wxTreebook()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`addPage(This::wxTreebook(), Page::wxWindow() (see module wxWindow), Text::string()) -> bool()`

Equivalent to `addPage(This, Page, Text, [])`.

`addPage(This::wxTreebook(), Page::wxWindow() (see module wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

**Option** = {bSelect, bool()} | {imageId, integer()}

See external documentation.

`advanceSelection(This::wxTreebook()) -> ok`

Equivalent to `advanceSelection(This, [])`.

`advanceSelection(This::wxTreebook(), Options::[Option]) -> ok`

Types:

**Option** = {forward, bool()}

See external documentation.

**assignImageList**(This::wxTreebook(), ImageList::wxImageList() (see module wxImageList)) -> ok

See external documentation.

**create**(This::wxTreebook(), Parent::wxWindow() (see module wxWindow), Id::integer()) -> bool()

Equivalent to *create(This, Parent, Id, [])*.

**create**(This::wxTreebook(), Parent::wxWindow() (see module wxWindow), Id::integer(), Options::[Option]) -> bool()

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

**deleteAllPages**(This::wxTreebook()) -> bool()

See external documentation.

**deletePage**(This::wxTreebook(), Pos::integer()) -> bool()

See external documentation.

**removePage**(This::wxTreebook(), N::integer()) -> bool()

See external documentation.

**getCurrentPage**(This::wxTreebook()) -> wxWindow() (see module wxWindow)

See external documentation.

**getImageList**(This::wxTreebook()) -> wxImageList() (see module wxImageList)

See external documentation.

**getPage**(This::wxTreebook(), N::integer()) -> wxWindow() (see module wxWindow)

See external documentation.

**getPageCount**(This::wxTreebook()) -> integer()

See external documentation.

**getPageImage**(This::wxTreebook(), N::integer()) -> integer()

See external documentation.

**getPageText**(This::wxTreebook(), N::integer()) -> string()

See external documentation.

`getSelection(This::wxTreebook()) -> integer()`

See [external documentation](#).

`expandNode(This::wxTreebook(), Pos::integer()) -> bool()`

Equivalent to `expandNode(This, Pos, [])`.

`expandNode(This::wxTreebook(), Pos::integer(), Options::[Option]) -> bool()`

Types:

**Option** = {expand, bool()}

See [external documentation](#).

`isNodeExpanded(This::wxTreebook(), Pos::integer()) -> bool()`

See [external documentation](#).

`hitTest(This::wxTreebook(), Pt::{X::integer(), Y::integer()}) -> {integer(),  
Flags::integer()}`

See [external documentation](#).

`insertPage(This::wxTreebook(), Pos::integer(), Page::wxWindow() (see module  
wxWindow), Text::string()) -> bool()`

Equivalent to `insertPage(This, Pos, Page, Text, [])`.

`insertPage(This::wxTreebook(), Pos::integer(), Page::wxWindow() (see module  
wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

**Option** = {bSelect, bool()} | {imageId, integer()}

See [external documentation](#).

`insertSubPage(This::wxTreebook(), Pos::integer(), Page::wxWindow() (see  
module wxWindow), Text::string()) -> bool()`

Equivalent to `insertSubPage(This, Pos, Page, Text, [])`.

`insertSubPage(This::wxTreebook(), Pos::integer(), Page::wxWindow() (see  
module wxWindow), Text::string(), Options::[Option]) -> bool()`

Types:

**Option** = {bSelect, bool()} | {imageId, integer()}

See [external documentation](#).

`setImageList(This::wxTreebook(), ImageList::wxImageList() (see module  
wxImageList)) -> ok`

See [external documentation](#).

**setSize(This::wxTreebook(), Size::{W::integer(), H::integer()}) -> ok**

See external documentation.

**setImage(This::wxTreebook(), N::integer(), ImageId::integer()) -> bool()**

See external documentation.

**setText(This::wxTreebook(), N::integer(), StrText::string()) -> bool()**

See external documentation.

**getSelection(This::wxTreebook(), N::integer()) -> integer()**

See external documentation.

**changeSelection(This::wxTreebook(), N::integer()) -> integer()**

See external documentation.

**destroy(This::wxTreebook()) -> ok**

Destroys this object, do not use object again

## wxUpdateUIEvent

---

Erlang module

See external documentation: **wxUpdateUIEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*update\_ui*

See also the message variant *#wxUpdateUI{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

## DATA TYPES

`wxUpdateUIEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`canUpdate(Win::wxWindow() (see module wxWindow)) -> bool()`

See external documentation.

`check(This::wxUpdateUIEvent(), Check::bool()) -> ok`

See external documentation.

`enable(This::wxUpdateUIEvent(), Enable::bool()) -> ok`

See external documentation.

`show(This::wxUpdateUIEvent(), Show::bool()) -> ok`

See external documentation.

`getChecked(This::wxUpdateUIEvent()) -> bool()`

See external documentation.

`getEnabled(This::wxUpdateUIEvent()) -> bool()`

See external documentation.

`getShown(This::wxUpdateUIEvent()) -> bool()`

See external documentation.

`getSetChecked(This::wxUpdateUIEvent()) -> bool()`

See external documentation.



**getSetEnabled(This::wxUpdateUIEvent()) -> bool()**

See external documentation.

**getSetShown(This::wxUpdateUIEvent()) -> bool()**

See external documentation.

**getSetText(This::wxUpdateUIEvent()) -> bool()**

See external documentation.

**getText(This::wxUpdateUIEvent()) -> string()**

See external documentation.

**getMode() -> WxUpdateUIMode**

Types:

**WxUpdateUIMode = integer()**

See external documentation.

WxUpdateUIMode is one of ?wxUPDATE\_UI\_PROCESS\_ALL | ?wxUPDATE\_UI\_PROCESS\_SPECIFIED

**getUpdateInterval() -> integer()**

See external documentation.

**resetUpdateTime() -> ok**

See external documentation.

**setMode(Mode::WxUpdateUIMode) -> ok**

Types:

**WxUpdateUIMode = integer()**

See external documentation.

WxUpdateUIMode is one of ?wxUPDATE\_UI\_PROCESS\_ALL | ?wxUPDATE\_UI\_PROCESS\_SPECIFIED

**setText(This::wxUpdateUIEvent(), Text::string()) -> ok**

See external documentation.

**setUpdateInterval(UpdateInterval::integer()) -> ok**

See external documentation.

## wxWindow

---

Erlang module

See external documentation: **wxWindow**.

This class is derived (and can use functions) from:  
*wxEvtHandler*

## DATA TYPES

`wxWindow()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## Exports

`new() -> wxWindow()`

See external documentation.

`new(Parent::wxWindow(), Id::integer()) -> wxWindow()`

Equivalent to `new(Parent, Id, [])`.

`new(Parent::wxWindow(), Id::integer(), Options::[Option]) -> wxWindow()`

Types:

**Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`cacheBestSize(This::wxWindow(), Size::{W::integer(), H::integer()}) -> ok`

See external documentation.

`captureMouse(This::wxWindow()) -> ok`

See external documentation.

`center(This::wxWindow()) -> ok`

Equivalent to `center(This, [])`.

`center(This::wxWindow(), Options::[Option]) -> ok`

Types:

**Option** = {dir, integer()}

See external documentation.

`centerOnParent(This::wxWindow()) -> ok`

Equivalent to `centerOnParent(This, [])`.

`centerOnParent(This::wxWindow(), Options::[Option]) -> ok`

Types:

**Option** = {dir, integer()}

See external documentation.

`centre(This::wxWindow()) -> ok`

Equivalent to `centre(This, [])`.

`centre(This::wxWindow(), Options::[Option]) -> ok`

Types:

**Option** = {dir, integer()}

See external documentation.

`centreOnParent(This::wxWindow()) -> ok`

Equivalent to `centreOnParent(This, [])`.

`centreOnParent(This::wxWindow(), Options::[Option]) -> ok`

Types:

**Option** = {dir, integer()}

See external documentation.

`clearBackground(This::wxWindow()) -> ok`

See external documentation.

`clientToScreen(This::wxWindow(), Pt::{X::integer(), Y::integer()}) -> {X::integer(), Y::integer()}`

See external documentation.

`clientToScreen(This::wxWindow(), X::integer(), Y::integer()) -> {X::integer(), Y::integer()}`

See external documentation.

`close(This::wxWindow()) -> bool()`

Equivalent to `close(This, [])`.

`close(This::wxWindow(), Options::[Option]) -> bool()`

Types:

**Option** = {force, bool()}

See external documentation.

`convertDialogToPixels(This::wxWindow(), Sz::{W::integer(), H::integer()}) -> {W::integer(), H::integer()}`

See external documentation.

`convertPixelsToDialog(This::wxWindow(), Sz::{W::integer(), H::integer()}) -> {W::integer(), H::integer()}`

See [external documentation](#).

`Destroy(This::wxWindow()) -> bool()`

See [external documentation](#).

`destroyChildren(This::wxWindow()) -> bool()`

See [external documentation](#).

`disable(This::wxWindow()) -> bool()`

See [external documentation](#).

`enable(This::wxWindow()) -> bool()`

Equivalent to `enable(This, [])`.

`enable(This::wxWindow(), Options::[Option]) -> bool()`

Types:

**Option** = {enable, bool()}

See [external documentation](#).

`findFocus() -> wxWindow()`

See [external documentation](#).

`findWindow(This::wxWindow(), X::integer() | string()) -> wxWindow()`

See [external documentation](#).

Alternatives:

`findWindow(This::wxWindow(), Winid::integer()) -> wxWindow()`

`findWindow(This::wxWindow(), Name::string()) -> wxWindow()`

`findWindowById(Winid::integer()) -> wxWindow()`

Equivalent to `findWindowById(Winid, [])`.

`findWindowById(Winid::integer(), Options::[Option]) -> wxWindow()`

Types:

**Option** = {parent, wxWindow()}

See [external documentation](#).

`findWindowByName(Name::string()) -> wxWindow()`

Equivalent to `findWindowByName(Name, [])`.

`findWindowByName(Name::string(), Options::[Option]) -> wxWindow()`

Types:

**Option** = {parent, wxWindow()}

See external documentation.

**findWindowByLabel**(Label::string()) -> wxWindow()

Equivalent to *findWindowByLabel*(Label, []).

**findWindowByLabel**(Label::string(), Options::[Option]) -> wxWindow()

Types:

**Option** = {parent, wxWindow()}

See external documentation.

**fit**(This::wxWindow()) -> ok

See external documentation.

**fitInside**(This::wxWindow()) -> ok

See external documentation.

**freeze**(This::wxWindow()) -> ok

See external documentation.

**getAcceleratorTable**(This::wxWindow()) -> wxAcceleratorTable() (see module wxAcceleratorTable)

See external documentation.

**getBackgroundColour**(This::wxWindow()) -> colour() (see module wx)

See external documentation.

**getBackgroundStyle**(This::wxWindow()) -> WxBackgroundStyle

Types:

**WxBackgroundStyle** = integer()

See external documentation.

WxBackgroundStyle is one of ?wxBG\_STYLE\_SYSTEM | ?wxBG\_STYLE\_COLOUR | ?wxBG\_STYLE\_CUSTOM

**getBestSize**(This::wxWindow()) -> {W::integer(), H::integer()}

See external documentation.

**getCaret**(This::wxWindow()) -> wxCaret() (see module wxCaret)

See external documentation.

**getCapture**() -> wxWindow()

See external documentation.

`getCharHeight(This::wxWindow()) -> integer()`

See external documentation.

`getCharWidth(This::wxWindow()) -> integer()`

See external documentation.

`getChildren(This::wxWindow()) -> [wxWindow()]`

See external documentation.

`getClientSize(This::wxWindow()) -> {W::integer(), H::integer()}`

See external documentation.

`getContainingSizer(This::wxWindow()) -> wxSizer() (see module wxSizer)`

See external documentation.

`getCursor(This::wxWindow()) -> wxCursor() (see module wxCursor)`

See external documentation.

`getDropTarget(This::wxWindow()) -> wxDropTarget() (see module wxDropTarget)`

See external documentation.

`getEventHandler(This::wxWindow()) -> wxEvtHandler() (see module wxEvtHandler)`

See external documentation.

`getExtraStyle(This::wxWindow()) -> integer()`

See external documentation.

`getFont(This::wxWindow()) -> wxFont() (see module wxFont)`

See external documentation.

`getForegroundColour(This::wxWindow()) -> colour() (see module wx)`

See external documentation.

`getGrandParent(This::wxWindow()) -> wxWindow()`

See external documentation.

`getHandle(This::wxWindow()) -> integer()`

See external documentation.

`getHelpText(This::wxWindow()) -> string()`

See external documentation.

`getId(This::wxWindow()) -> integer()`

See external documentation.

`getLabel(This::wxWindow()) -> string()`

See external documentation.

`getMaxSize(This::wxWindow()) -> {W::integer(), H::integer()}`

See external documentation.

`getMinSize(This::wxWindow()) -> {W::integer(), H::integer()}`

See external documentation.

`getName(This::wxWindow()) -> string()`

See external documentation.

`getParent(This::wxWindow()) -> wxWindow()`

See external documentation.

`getPosition(This::wxWindow()) -> {X::integer(), Y::integer()}`

See external documentation.

`getRect(This::wxWindow()) -> {X::integer(), Y::integer(), W::integer(),  
H::integer()}`

See external documentation.

`getScreenPosition(This::wxWindow()) -> {X::integer(), Y::integer()}`

See external documentation.

`getScreenRect(This::wxWindow()) -> {X::integer(), Y::integer(), W::integer(),  
H::integer()}`

See external documentation.

`getScrollPos(This::wxWindow(), Orient::integer()) -> integer()`

See external documentation.

`getScrollRange(This::wxWindow(), Orient::integer()) -> integer()`

See external documentation.

`getScrollThumb(This::wxWindow(), Orient::integer()) -> integer()`

See external documentation.

`getSize(This::wxWindow()) -> {W::integer(), H::integer()}`

See external documentation.

`getSizer(This::wxWindow()) -> wxSizer()` (see module `wxSizer`)

See external documentation.

`getTextExtent(This::wxWindow(), String::string()) -> {X::integer(),  
Y::integer(), Descent::integer(), ExternalLeading::integer()}`

Equivalent to `getTextExtent(This, String, [])`.

`getTextExtent(This::wxWindow(), String::string(), Options::[Option]) ->  
{X::integer(), Y::integer(), Descent::integer(), ExternalLeading::integer()}`

Types:

**Option** = {theFont, wxFont() (see module `wxFont`)}

See external documentation.

`getToolTip(This::wxWindow()) -> wxToolTip()` (see module `wxToolTip`)

See external documentation.

`getUpdateRegion(This::wxWindow()) -> wxRegion()` (see module `wxRegion`)

See external documentation.

`getVirtualSize(This::wxWindow()) -> {W::integer(), H::integer()}`

See external documentation.

`getWindowStyleFlag(This::wxWindow()) -> integer()`

See external documentation.

`getWindowVariant(This::wxWindow()) -> WxWindowVariant`

Types:

**WxWindowVariant** = integer()

See external documentation.

WxWindowVariant is one of ?wxWINDOW\_VARIANT\_NORMAL | ?wxWINDOW\_VARIANT\_SMALL | ?  
wxWINDOW\_VARIANT\_MINI | ?wxWINDOW\_VARIANT\_LARGE | ?wxWINDOW\_VARIANT\_MAX

`hasCapture(This::wxWindow()) -> bool()`

See external documentation.

`hasScrollbar(This::wxWindow(), Orient::integer()) -> bool()`

See external documentation.

`hasTransparentBackground(This::wxWindow()) -> bool()`

See external documentation.

`hide(This::wxWindow()) -> bool()`

See external documentation.



**inheritAttributes(This::wxWindow()) -> ok**

See external documentation.

**initDialog(This::wxWindow()) -> ok**

See external documentation.

**invalidateBestSize(This::wxWindow()) -> ok**

See external documentation.

**isEnabled(This::wxWindow()) -> bool()**

See external documentation.

**isExposed(This::wxWindow(), X::term()) -> bool()**

See external documentation.

Alternatives:

**isExposed(This::wxWindow(), Pt::{X::integer(),Y::integer()}) -> bool()**

**isExposed(This::wxWindow(), Rect::{X::integer(),Y::integer(),W::integer(),H::integer()}) -> bool()**

**isExposed(This::wxWindow(), X::integer(), Y::integer()) -> bool()**

See external documentation.

**isExposed(This::wxWindow(), X::integer(), Y::integer(), W::integer(), H::integer()) -> bool()**

See external documentation.

**isRetained(This::wxWindow()) -> bool()**

See external documentation.

**isShown(This::wxWindow()) -> bool()**

See external documentation.

**isTopLevel(This::wxWindow()) -> bool()**

See external documentation.

**layout(This::wxWindow()) -> bool()**

See external documentation.

**lineDown(This::wxWindow()) -> bool()**

See external documentation.

**lineUp(This::wxWindow()) -> bool()**

See external documentation.

`lower(This::wxWindow()) -> ok`

See [external documentation](#).

`makeModal(This::wxWindow()) -> ok`

Equivalent to `makeModal(This, [])`.

`makeModal(This::wxWindow(), Options::[Option]) -> ok`

Types:

**Option** = {modal, bool()}

See [external documentation](#).

`move(This::wxWindow(), Pt::{X::integer(), Y::integer()}) -> ok`

Equivalent to `move(This, Pt, [])`.

`move(This::wxWindow(), X::integer() | term(), X::integer() | term()) -> ok`

See [external documentation](#).

Alternatives:

`move(This::wxWindow(), X::integer(), Y::integer()) -> move(This, X, Y, [])`

`move(This::wxWindow(), Pt::{X::integer(), Y::integer()}, [Option]) -> ok`

**Option** = {flags, integer()}

`move(This::wxWindow(), X::integer(), Y::integer(), Options::[Option]) -> ok`

Types:

**Option** = {flags, integer()}

See [external documentation](#).

`moveAfterInTabOrder(This::wxWindow(), Win::wxWindow()) -> ok`

See [external documentation](#).

`moveBeforeInTabOrder(This::wxWindow(), Win::wxWindow()) -> ok`

See [external documentation](#).

`navigate(This::wxWindow()) -> bool()`

Equivalent to `navigate(This, [])`.

`navigate(This::wxWindow(), Options::[Option]) -> bool()`

Types:

**Option** = {flags, integer()}

See [external documentation](#).

`pageDown(This::wxWindow()) -> bool()`

See [external documentation](#).

**pageUp(This::wxWindow()) -> bool()**

See external documentation.

**popEventHandler(This::wxWindow()) -> wxEvtHandler()** (see module wxEvtHandler)

Equivalent to *popEventHandler(This, [])*.

**popEventHandler(This::wxWindow(), Options::[Option]) -> wxEvtHandler()** (see module wxEvtHandler)

Types:

**Option = {deleteHandler, bool()}**

See external documentation.

**popupMenu(This::wxWindow(), Menu::wxMenu())** (see module wxMenu) -> bool()

Equivalent to *popupMenu(This, Menu, [])*.

**popupMenu(This::wxWindow(), Menu::wxMenu())** (see module wxMenu), Options::[Option]) -> bool()

Types:

**Option = {pos, {X::integer(), Y::integer()}}**

See external documentation.

**popupMenu(This::wxWindow(), Menu::wxMenu())** (see module wxMenu), X::integer(), Y::integer()) -> bool()

See external documentation.

**raise(This::wxWindow()) -> ok**

See external documentation.

**refresh(This::wxWindow()) -> ok**

Equivalent to *refresh(This, [])*.

**refresh(This::wxWindow(), Options::[Option]) -> ok**

Types:

**Option = {eraseBackground, bool()} | {rect, {X::integer(), Y::integer(), W::integer(), H::integer()}}**

See external documentation.

**refreshRect(This::wxWindow(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}) -> ok**

Equivalent to *refreshRect(This, Rect, [])*.

**refreshRect(This::wxWindow(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()})**, Options::[Option]) -> ok

Types:

**Option** = {eraseBackground, bool()}

See external documentation.

**releaseMouse**(This::wxWindow()) -> ok

See external documentation.

**removeChild**(This::wxWindow(), Child::wxWindow()) -> ok

See external documentation.

**reparent**(This::wxWindow(), NewParent::wxWindow()) -> bool()

See external documentation.

**screenToClient**(This::wxWindow()) -> {X::integer(), Y::integer()}

See external documentation.

**screenToClient**(This::wxWindow(), Pt::{X::integer(), Y::integer()}) -> {X::integer(), Y::integer()}

See external documentation.

**scrollLines**(This::wxWindow(), Lines::integer()) -> bool()

See external documentation.

**scrollPages**(This::wxWindow(), Pages::integer()) -> bool()

See external documentation.

**scrollWindow**(This::wxWindow(), Dx::integer(), Dy::integer()) -> ok

Equivalent to *scrollWindow(This, Dx, Dy, [])*.

**scrollWindow**(This::wxWindow(), Dx::integer(), Dy::integer(), Options::[Option]) -> ok

Types:

**Option** = {rect, {X::integer(), Y::integer(), W::integer(), H::integer()}}

See external documentation.

**setAcceleratorTable**(This::wxWindow(), Accel::wxAcceleratorTable() (see module wxAcceleratorTable)) -> ok

See external documentation.

**setAutoLayout**(This::wxWindow(), AutoLayout::bool()) -> ok

See external documentation.

```
setBackgroundColour(This::wxWindow(), Colour::colour() (see module wx)) ->
bool()
```

See external documentation.

```
setBackgroundStyle(This::wxWindow(), Style::WxBackgroundStyle) -> bool()
```

Types:

**WxBackgroundStyle** = integer()

See external documentation.

WxBackgroundStyle is one of ?wxBG\_STYLE\_SYSTEM | ?wxBG\_STYLE\_COLOUR | ?wxBG\_STYLE\_CUSTOM

```
setCaret(This::wxWindow(), Caret::wxCaret() (see module wxCaret)) -> ok
```

See external documentation.

```
setClientSize(This::wxWindow(), X::term()) -> ok
```

See external documentation.

Alternatives:

```
setClientSize(This::wxWindow(), Size::{W::integer(),H::integer()}) -> ok
```

```
setClientSize(This::wxWindow(), Rect::{X::integer(),Y::integer(),W::integer(),H::integer()}) -> ok
```

```
setClientSize(This::wxWindow(), Width::integer(), Height::integer()) -> ok
```

See external documentation.

```
setContainingSizer(This::wxWindow(), Sizer::wxSizer() (see module wxSizer)) -> ok
```

See external documentation.

```
setCursor(This::wxWindow(), Cursor::wxCursor() (see module wxCursor)) ->
bool()
```

See external documentation.

```
setMaxSize(This::wxWindow(), MaxSize::{W::integer(), H::integer()}) -> ok
```

See external documentation.

```
setMinSize(This::wxWindow(), MinSize::{W::integer(), H::integer()}) -> ok
```

See external documentation.

```
setOwnBackgroundColour(This::wxWindow(), Colour::colour() (see module wx)) ->
ok
```

See external documentation.

```
setOwnFont(This::wxWindow(), Font::wxFont() (see module wxFont)) -> ok
```

See external documentation.

`setOwnForegroundColour(This::wxWindow(), Colour::colour() (see module wx)) -> ok`

See external documentation.

`setDropTarget(This::wxWindow(), DropTarget::wxDropTarget() (see module wxDropTarget)) -> ok`

See external documentation.

`setExtraStyle(This::wxWindow(), ExStyle::integer()) -> ok`

See external documentation.

`setFocus(This::wxWindow()) -> ok`

See external documentation.

`setFocusFromKbd(This::wxWindow()) -> ok`

See external documentation.

`setFont(This::wxWindow(), Font::wxFont() (see module wxFont)) -> bool()`

See external documentation.

`setForegroundColour(This::wxWindow(), Colour::colour() (see module wx)) -> bool()`

See external documentation.

`setHelpText(This::wxWindow(), Text::string()) -> ok`

See external documentation.

`setId(This::wxWindow(), Winid::integer()) -> ok`

See external documentation.

`setLabel(This::wxWindow(), Label::string()) -> ok`

See external documentation.

`setName(This::wxWindow(), Name::string()) -> ok`

See external documentation.

`setPalette(This::wxWindow(), Pal::wxPalette() (see module wxPalette)) -> ok`

See external documentation.

`setScrollbar(This::wxWindow(), Orient::integer(), Pos::integer(), ThumbVisible::integer(), Range::integer()) -> ok`

Equivalent to `setScrollbar(This, Orient, Pos, ThumbVisible, Range, [])`.

```
setScrollbar(This::wxWindow(), Orient::integer(), Pos::integer(),
ThumbVisible::integer(), Range::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {refresh, bool()}

See external documentation.

```
setScrollPos(This::wxWindow(), Orient::integer(), Pos::integer()) -> ok
```

Equivalent to *setScrollPos(This, Orient, Pos, [])*.

```
setScrollPos(This::wxWindow(), Orient::integer(), Pos::integer(), Options::
[Option]) -> ok
```

Types:

**Option** = {refresh, bool()}

See external documentation.

```
setSize(This::wxWindow(), X::term()) -> ok
```

See external documentation.

Alternatives:

```
setSize(This::wxWindow(), Rect::
{X::integer(),Y::integer(),W::integer(),H::integer()}) -> setSize(This,Rect,
[])
```

```
setSize(This::wxWindow(), Size::{W::integer(),H::integer()}) -> ok
```

```
setSize(This::wxWindow(), X::integer() | term(), X::integer() | term()) -> ok
```

See external documentation.

Alternatives:

```
setSize(This::wxWindow(), Width::integer(), Height::integer()) -> ok
```

```
setSize(This::wxWindow(), Rect::
{X::integer(),Y::integer(),W::integer(),H::integer()}, [Option]) -> ok
Option = {sizeFlags, integer()}
```

```
setSize(This::wxWindow(), X::integer(), Y::integer(), Width::integer(),
Height::integer()) -> ok
```

Equivalent to *setSize(This, X, Y, Width, Height, [])*.

```
setSize(This::wxWindow(), X::integer(), Y::integer(), Width::integer(),
Height::integer(), Options::[Option]) -> ok
```

Types:

**Option** = {sizeFlags, integer()}

See external documentation.

```
setSizeHints(This::wxWindow(), MinSize::{W::integer(), H::integer()}) -> ok
```

Equivalent to *setSizeHints(This, MinSize, [])*.

```
setSizeHints(This::wxWindow(), X::integer() | term(), X::integer() | term())  
-> ok
```

See [external documentation](#).

Alternatives:

```
setSizeHints(This::wxWindow(), MinW::integer(), MinH::integer()) ->  
setSizeHints(This, MinW, MinH, [])
```

```
setSizeHints(This::wxWindow(), MinSize:: {W::integer(), H::integer()},  
[Option]) -> ok
```

```
Option = {maxSize, {W::integer(), H::integer()}} | {incSize, {W::integer(), H::integer()}}
```

```
setSizeHints(This::wxWindow(), MinW::integer(), MinH::integer(), Options::  
[Option]) -> ok
```

Types:

**Option** = {maxW, integer()} | {maxH, integer()} | {incW, integer()} | {incH, integer()}

See [external documentation](#).

```
setSize(This::wxWindow(),Sizer::wxSizer() (see module wxSizer)) -> ok
```

Equivalent to *setSize(This, Sizer, [])*.

```
setSize(This::wxWindow(),Sizer::wxSizer() (see module wxSizer),Options::  
[Option]) -> ok
```

Types:

**Option** = {deleteOld, bool()}

See [external documentation](#).

```
setSizeAndFit(This::wxWindow(),Sizer::wxSizer() (see module wxSizer)) -> ok
```

Equivalent to *setSizeAndFit(This, Sizer, [])*.

```
setSizeAndFit(This::wxWindow(),Sizer::wxSizer() (see module wxSizer),  
Options:: [Option]) -> ok
```

Types:

**Option** = {deleteOld, bool()}

See [external documentation](#).

```
setThemeEnabled(This::wxWindow(), EnableTheme::bool()) -> ok
```

See [external documentation](#).

```
setToolTip(This::wxWindow(), X::string() | term()) -> ok
```

See [external documentation](#).

Alternatives:

```
setToolTip(This::wxWindow(), Tip::string()) -> ok
```

```
setToolTip(This::wxWindow(), Tip::wxToolTip:wxToolTip()) -> ok
```



```
setVirtualSize(This::wxWindow(), Size::{W::integer(), H::integer()}) -> ok
```

See external documentation.

```
setVirtualSize(This::wxWindow(), X::integer(), Y::integer()) -> ok
```

See external documentation.

```
setVirtualSizeHints(This::wxWindow(), MinSize::{W::integer(), H::integer()})  
-> ok
```

Equivalent to *setVirtualSizeHints(This, MinSize, [])*.

```
setVirtualSizeHints(This::wxWindow(), X::integer() | term(), X::integer() |  
term()) -> ok
```

See external documentation.

Alternatives:

```
setVirtualSizeHints(This::wxWindow(), MinW::integer(), MinH::integer()) ->  
setVirtualSizeHints(This, MinW, MinH, [])
```

```
setVirtualSizeHints(This::wxWindow(), MinSize::{W::integer(), H::integer()},  
[Option]) -> ok
```

```
Option = {maxSize, {W::integer(), H::integer()}}
```

```
setVirtualSizeHints(This::wxWindow(), MinW::integer(), MinH::integer(),  
Options::[Option]) -> ok
```

Types:

```
Option = {maxW, integer()} | {maxH, integer()}
```

See external documentation.

```
setWindowStyle(This::wxWindow(), Style::integer()) -> ok
```

See external documentation.

```
setWindowStyleFlag(This::wxWindow(), Style::integer()) -> ok
```

See external documentation.

```
setWindowVariant(This::wxWindow(), Variant::WxWindowVariant) -> ok
```

Types:

```
WxWindowVariant = integer()
```

See external documentation.

WxWindowVariant is one of ?wxWINDOW\_VARIANT\_NORMAL | ?wxWINDOW\_VARIANT\_SMALL | ?  
wxWINDOW\_VARIANT\_MINI | ?wxWINDOW\_VARIANT\_LARGE | ?wxWINDOW\_VARIANT\_MAX

```
shouldInheritColours(This::wxWindow()) -> bool()
```

See external documentation.

```
show(This::wxWindow()) -> bool()
```

Equivalent to *show(This, [])*.

**show**(This::wxWindow(), Options::[Option]) -> bool()

Types:

Option = {show, bool()}

See external documentation.

**thaw**(This::wxWindow()) -> ok

See external documentation.

**transferDataFromWindow**(This::wxWindow()) -> bool()

See external documentation.

**transferDataToWindow**(This::wxWindow()) -> bool()

See external documentation.

**update**(This::wxWindow()) -> ok

See external documentation.

**updateWindowUI**(This::wxWindow()) -> ok

Equivalent to *updateWindowUI(This, [])*.

**updateWindowUI**(This::wxWindow(), Options::[Option]) -> ok

Types:

Option = {flags, integer()}

See external documentation.

**validate**(This::wxWindow()) -> bool()

See external documentation.

**warpPointer**(This::wxWindow(), X::integer(), Y::integer()) -> ok

See external documentation.

**destroy**(This::wxWindow()) -> ok

Destroys this object, do not use object again

## wxWindowCreateEvent

---

Erlang module

See external documentation: **wxWindowCreateEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*create*

See also the message variant *#wxWindowCreate{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

### DATA TYPES

`wxWindowCreateEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxWindowDC

---

Erlang module

See external documentation: **wxWindowDC**.

This class is derived (and can use functions) from:  
*wxDC*

### DATA TYPES

`wxWindowDC()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

`new()` -> `wxWindowDC()`

See **external documentation**.

`new(Win::wxWindow())` (see module `wxWindow`) -> `wxWindowDC()`

See **external documentation**.

`destroy(This::wxWindowDC())` -> `ok`

Destroys this object, do not use object again

## wxWindowDestroyEvent

---

Erlang module

See external documentation: **wxWindowDestroyEvent**.

Use *wxEvtHandler:connect/3* with EventType:

*destroy*

See also the message variant *#wxWindowDestroy{}* event record type.

This class is derived (and can use functions) from:

*wxCommandEvent*

*wxEvent*

### DATA TYPES

`wxWindowDestroyEvent()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

## wxXmlResource

---

Erlang module

See external documentation: **wxXmlResource**.

### DATA TYPES

`wxXmlResource()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**`new()` -> `wxXmlResource()`**

Equivalent to `new([])`.

**`new(Options::[Option])` -> `wxXmlResource()`**

Types:

**`Option = {flags, integer()} | {domain, string()}`**

See external documentation.

**`new(Filemask::string(), Options::[Option])` -> `wxXmlResource()`**

Types:

**`Option = {flags, integer()} | {domain, string()}`**

See external documentation.

**`attachUnknownControl(This::wxXmlResource(), Name::string(), Control::wxWindow() (see module wxWindow))` -> `bool()`**

Equivalent to `attachUnknownControl(This, Name, Control, [])`.

**`attachUnknownControl(This::wxXmlResource(), Name::string(), Control::wxWindow() (see module wxWindow), Options::[Option])` -> `bool()`**

Types:

**`Option = {parent, wxWindow() (see module wxWindow)}`**

See external documentation.

**`clearHandlers(This::wxXmlResource())` -> `ok`**

See external documentation.

**`compareVersion(This::wxXmlResource(), Major::integer(), Minor::integer(), Release::integer(), Revision::integer())` -> `integer()`**

See external documentation.

`get()` -> `wxXmlResource()`

See external documentation.

`getFlags(This::wxXmlResource())` -> `integer()`

See external documentation.

`getVersion(This::wxXmlResource())` -> `integer()`

See external documentation.

`getXRCID(Str_id::[string()])` -> `integer()`

Equivalent to `getXRCID(Str_id, [])`.

`getXRCID(Str_id::[string()], Options::[Option])` -> `integer()`

Types:

`Option = {value_if_not_found, integer()}`

See external documentation.

`initAllHandlers(This::wxXmlResource())` -> `ok`

See external documentation.

`load(This::wxXmlResource(), Filemask::string())` -> `bool()`

See external documentation.

`loadBitmap(This::wxXmlResource(), Name::string())` -> `wxBitmap()` (see module `wxBitmap`)

See external documentation.

`loadDialog(This::wxXmlResource(), Parent::wxWindow() (see module wxWindow), Name::string())` -> `wxDialog()` (see module `wxDialog`)

See external documentation.

`loadDialog(This::wxXmlResource(), Dlg::wxDialog() (see module wxDialog), Parent::wxWindow() (see module wxWindow), Name::string())` -> `bool()`

See external documentation.

`loadFrame(This::wxXmlResource(), Parent::wxWindow() (see module wxWindow), Name::string())` -> `wxFrame()` (see module `wxFrame`)

See external documentation.

`loadFrame(This::wxXmlResource(), Frame::wxFrame() (see module wxFrame), Parent::wxWindow() (see module wxWindow), Name::string())` -> `bool()`

See external documentation.

`loadIcon(This::wxXmlResource(), Name::string()) -> wxIcon()` (see module `wxIcon`)

See external documentation.

`loadMenu(This::wxXmlResource(), Name::string()) -> wxMenu()` (see module `wxMenu`)

See external documentation.

`loadMenuBar(This::wxXmlResource(), Name::string()) -> wxMenuBar()` (see module `wxMenuBar`)

See external documentation.

`loadMenuBar(This::wxXmlResource(), Parent::wxWindow() (see module wxWindow), Name::string()) -> wxMenuBar()` (see module `wxMenuBar`)

See external documentation.

`loadPanel(This::wxXmlResource(), Parent::wxWindow() (see module wxWindow), Name::string()) -> wxPanel()` (see module `wxPanel`)

See external documentation.

`loadPanel(This::wxXmlResource(), Panel::wxPanel() (see module wxPanel), Parent::wxWindow() (see module wxWindow), Name::string()) -> bool()`

See external documentation.

`loadToolBar(This::wxXmlResource(), Parent::wxWindow() (see module wxWindow), Name::string()) -> wxToolBar()` (see module `wxToolBar`)

See external documentation.

`set(Res::wxXmlResource()) -> wxXmlResource()`

See external documentation.

`setFlags(This::wxXmlResource(), Flags::integer()) -> ok`

See external documentation.

`unload(This::wxXmlResource(), Filename::string()) -> bool()`

See external documentation.

`xrcctrl(Window::wxWindow() (see module wxWindow), Name::string(), Type::atom()) -> wxObject()` (see module `wx`)

Looks up a control with Name in a window created with XML resources. You can use it to set/get values from controls. The object is type casted to *Type*. Example:

```
Xrc = wxXmlResource::get(),
Dlg = wxDialog::new(),
true = wxXmlResource::loadDialog(Xrc, Dlg, Frame, "controls_dialog"),
LCtrl = xrcctrl(Dlg, "controls_listctrl", wxListCtrl),
```



```
wxListCtrl:insertColumn(LCtrl, 0, "Name", [{ width, 200}]),
```

```
destroy(This::wxXmlResource()) -> ok
```

Destroys this object, do not use object again

## wx\_misc

---

Erlang module

See external documentation: **Misc**.

## Exports

**getKeyState(Key::WxKeyCode) -> bool()**

Types:

**WxKeyCode = integer()**

See **external documentation**.

WxKeyCode is one of ?WXX\_BACK | ?WXX\_TAB | ?WXX\_RETURN | ?WXX\_ESCAPE | ?WXX\_SPACE | ?WXX\_DELETE | ?WXX\_START | ?WXX\_LBUTTON | ?WXX\_RBUTTON | ?WXX\_CANCEL | ?WXX\_MBUTTON | ?WXX\_CLEAR | ?WXX\_SHIFT | ?WXX\_ALT | ?WXX\_CONTROL | ?WXX\_MENU | ?WXX\_PAUSE | ?WXX\_CAPITAL | ?WXX\_END | ?WXX\_HOME | ?WXX\_LEFT | ?WXX\_UP | ?WXX\_RIGHT | ?WXX\_DOWN | ?WXX\_SELECT | ?WXX\_PRINT | ?WXX\_EXECUTE | ?WXX\_SNAPSHOT | ?WXX\_INSERT | ?WXX\_HELP | ?WXX\_NUMPAD0 | ?WXX\_NUMPAD1 | ?WXX\_NUMPAD2 | ?WXX\_NUMPAD3 | ?WXX\_NUMPAD4 | ?WXX\_NUMPAD5 | ?WXX\_NUMPAD6 | ?WXX\_NUMPAD7 | ?WXX\_NUMPAD8 | ?WXX\_NUMPAD9 | ?WXX\_MULTIPLY | ?WXX\_ADD | ?WXX\_SEPARATOR | ?WXX\_SUBTRACT | ?WXX\_DECIMAL | ?WXX\_DIVIDE | ?WXX\_F1 | ?WXX\_F2 | ?WXX\_F3 | ?WXX\_F4 | ?WXX\_F5 | ?WXX\_F6 | ?WXX\_F7 | ?WXX\_F8 | ?WXX\_F9 | ?WXX\_F10 | ?WXX\_F11 | ?WXX\_F12 | ?WXX\_F13 | ?WXX\_F14 | ?WXX\_F15 | ?WXX\_F16 | ?WXX\_F17 | ?WXX\_F18 | ?WXX\_F19 | ?WXX\_F20 | ?WXX\_F21 | ?WXX\_F22 | ?WXX\_F23 | ?WXX\_F24 | ?WXX\_NUMLOCK | ?WXX\_SCROLL | ?WXX\_PAGEUP | ?WXX\_PAGEDOWN | ?WXX\_NUMPAD\_SPACE | ?WXX\_NUMPAD\_TAB | ?WXX\_NUMPAD\_ENTER | ?WXX\_NUMPAD\_F1 | ?WXX\_NUMPAD\_F2 | ?WXX\_NUMPAD\_F3 | ?WXX\_NUMPAD\_F4 | ?WXX\_NUMPAD\_HOME | ?WXX\_NUMPAD\_LEFT | ?WXX\_NUMPAD\_UP | ?WXX\_NUMPAD\_RIGHT | ?WXX\_NUMPAD\_DOWN | ?WXX\_NUMPAD\_PAGEUP | ?WXX\_NUMPAD\_PAGEDOWN | ?WXX\_NUMPAD\_END | ?WXX\_NUMPAD\_BEGIN | ?WXX\_NUMPAD\_INSERT | ?WXX\_NUMPAD\_DELETE | ?WXX\_NUMPAD\_EQUAL | ?WXX\_NUMPAD\_MULTIPLY | ?WXX\_NUMPAD\_ADD | ?WXX\_NUMPAD\_SEPARATOR | ?WXX\_NUMPAD\_SUBTRACT | ?WXX\_NUMPAD\_DECIMAL | ?WXX\_NUMPAD\_DIVIDE | ?WXX\_WINDOWS\_LEFT | ?WXX\_WINDOWS\_RIGHT | ?WXX\_WINDOWS\_MENU | ?WXX\_COMMAND | ?WXX\_SPECIAL1 | ?WXX\_SPECIAL2 | ?WXX\_SPECIAL3 | ?WXX\_SPECIAL4 | ?WXX\_SPECIAL5 | ?WXX\_SPECIAL6 | ?WXX\_SPECIAL7 | ?WXX\_SPECIAL8 | ?WXX\_SPECIAL9 | ?WXX\_SPECIAL10 | ?WXX\_SPECIAL11 | ?WXX\_SPECIAL12 | ?WXX\_SPECIAL13 | ?WXX\_SPECIAL14 | ?WXX\_SPECIAL15 | ?WXX\_SPECIAL16 | ?WXX\_SPECIAL17 | ?WXX\_SPECIAL18 | ?WXX\_SPECIAL19 | ?WXX\_SPECIAL20

**getMousePosition() -> {X::integer(), Y::integer()}**

See **external documentation**.

**getMouseState() -> wxMouseState()** (see module wx)

See **external documentation**.

**setDetectableAutoRepeat(Flag::bool()) -> bool()**

See **external documentation**.

**bell()** -> ok

See external documentation.

**findMenuItemId**(Frame::wxFrame() (see module wxFrame), MenuString::string(), ItemString::string()) -> integer()

See external documentation.

**genericFindWindowAtPoint**(Pt::{X::integer(), Y::integer()}) -> wxWindow() (see module wxWindow)

See external documentation.

**findWindowAtPoint**(Pt::{X::integer(), Y::integer()}) -> wxWindow() (see module wxWindow)

See external documentation.

**beginBusyCursor()** -> ok

Equivalent to *beginBusyCursor([])*.

**beginBusyCursor**(Options::[Option]) -> ok

Types:

Option = {cursor, wxCursor() (see module wxCursor)}

See external documentation.

**endBusyCursor()** -> ok

See external documentation.

**isBusy()** -> bool()

See external documentation.

**shutdown**(WFlags::WxShutdownFlags) -> bool()

Types:

WxShutdownFlags = integer()

See external documentation.

WxShutdownFlags is one of ?wxSHUTDOWN\_POWEROFF | ?wxSHUTDOWN\_REBOOT

**shell()** -> bool()

Equivalent to *shell([])*.

**shell**(Options::[Option]) -> bool()

Types:

Option = {command, string() }

See external documentation.

**launchDefaultBrowser**(Url::string()) -> bool()

Equivalent to *launchDefaultBrowser*(Url, []).

**launchDefaultBrowser**(Url::string(), Options::[Option]) -> bool()

Types:

**Option** = {flags, integer()}

See external documentation.

**getEmailAddress**() -> string()

See external documentation.

**getUserId**() -> string()

See external documentation.

**getHomeDir**() -> string()

See external documentation.

**newId**() -> integer()

See external documentation.

**registerId**(Id::integer()) -> ok

See external documentation.

**getCurrentId**() -> integer()

See external documentation.

**getOsDescription**() -> string()

See external documentation.

**isPlatformLittleEndian**() -> bool()

See external documentation.

**isPlatform64Bit**() -> bool()

See external documentation.

## glu

---

Erlang module

A part of the standard OpenGL Utility api. See [www.opengl.org](http://www.opengl.org)

Booleans are represented by integers 0 and 1.

### DATA TYPES

`clamp()`

A float clamped between 0.0 - 1.0

`enum()`

An integer defined in `gl.hrl`

`offset()`

An integer which is an offset in an array

`wx_mem()`

see `wx.erl` on memory allocation functions

### Exports

`tessellate(X1::Vec3, Vs::[Vec3]) -> {Triangles, VertexPos}`

Types:

`Vec3 = {float(), float(), float()}`

`Triangles = [VertexIndex::integer()]`

`VertexPos = binary()`

General purpose polygon triangulation. The first argument is the normal and the second a list of vertex positions. Returned is a list of indecies of the vertices and a binary (64bit native float) containing an array of vertex positions, it starts with the vertices in `Vs` and may contain newly created vertices in the end.

`build1DMipmapLevels(Target::enum(), InternalFormat::integer(), Width::integer(), Format::enum(), Type::enum(), Level::integer(), Base::integer(), Max::integer(), Data::binary()) -> integer()`

See **external** documentation.

`build1DMipmaps(Target::enum(), InternalFormat::integer(), Width::integer(), Format::enum(), Type::enum(), Data::binary()) -> integer()`

See **external** documentation.

`build2DMipmapLevels(Target::enum(), InternalFormat::integer(), Width::integer(), Height::integer(), Format::enum(), Type::enum(), Level::integer(), Base::integer(), Max::integer(), Data::binary()) -> integer()`

See **external** documentation.

```
build2DMipmaps(Target::enum(), InternalFormat::integer(), Width::integer(),  
Height::integer(), Format::enum(), Type::enum(), Data::binary()) -> integer()
```

See **external** documentation.

```
build3DMipmapLevels(Target::enum(), InternalFormat::integer(),  
Width::integer(), Height::integer(), Depth::integer(), Format::enum(),  
Type::enum(), Level::integer(), Base::integer(), Max::integer(),  
Data::binary()) -> integer()
```

See **external** documentation.

```
build3DMipmaps(Target::enum(), InternalFormat::integer(), Width::integer(),  
Height::integer(), Depth::integer(), Format::enum(), Type::enum(),  
Data::binary()) -> integer()
```

See **external** documentation.

```
checkExtension(ExtName::[integer()], ExtString::[integer()]) -> 0 | 1
```

See **external** documentation.

```
cylinder(Quad::integer(), Base::float(), Top::float(), Height::float(),  
Slices::integer(), Stacks::integer()) -> ok
```

See **external** documentation.

```
deleteQuadric(Quad::integer()) -> ok
```

See **external** documentation.

```
disk(Quad::integer(), Inner::float(), Outer::float(), Slices::integer(),  
Loops::integer()) -> ok
```

See **external** documentation.

```
errorString(Error::enum()) -> string()
```

See **external** documentation.

```
getString(Name::enum()) -> string()
```

See **external** documentation.

```
lookAt(EyeX::float(), EyeY::float(), EyeZ::float(), CenterX::float(),  
CenterY::float(), CenterZ::float(), UpX::float(), UpY::float(), UpZ::float())  
-> ok
```

See **external** documentation.

```
newQuadric() -> integer()
```

See **external** documentation.

`ortho2D(Left::float(), Right::float(), Bottom::float(), Top::float()) -> ok`

See **external** documentation.

`partialDisk(Quad::integer(), Inner::float(), Outer::float(),  
Slices::integer(), Loops::integer(), Start::float(), Sweep::float()) -> ok`

See **external** documentation.

`perspective(Fovy::float(), Aspect::float(), ZNear::float(), ZFar::float()) ->  
ok`

See **external** documentation.

`pickMatrix(X::float(), Y::float(), DelX::float(), DelY::float(), Viewport::  
{integer()}) -> ok`

See **external** documentation.

`project(ObjX::float(), ObjY::float(), ObjZ::float(), Model::{float()}, Proj::  
{float()}, View::{integer()}) -> {integer(), WinX::float(), WinY::float(),  
WinZ::float()}`

See **external** documentation.

`quadricDrawStyle(Quad::integer(), Draw::enum()) -> ok`

See **external** documentation.

`quadricNormals(Quad::integer(), Normal::enum()) -> ok`

See **external** documentation.

`quadricOrientation(Quad::integer(), Orientation::enum()) -> ok`

See **external** documentation.

`quadricTexture(Quad::integer(), Texture::0 | 1) -> ok`

See **external** documentation.

`scaleImage(Format::enum(), WIn::integer(), HIn::integer(), TypeIn::enum(),  
DataIn::binary(), WOut::integer(), HOut::integer(), TypeOut::enum(),  
DataOut::wx_mem() (see module wx)) -> integer()`

See **external** documentation.

`sphere(Quad::integer(), Radius::float(), Slices::integer(),  
Stacks::integer()) -> ok`

See **external** documentation.

```
unProject(WinX::float(), WinY::float(), WinZ::float(), Model::{float()},  
Proj::{float()}, View::{integer()}) -> {integer(), ObjX::float(),  
ObjY::float(), ObjZ::float()}
```

See **external** documentation.

```
unProject4(WinX::float(), WinY::float(), WinZ::float(), ClipW::float(),  
Model::{float()}, Proj::{float()}, View::{integer()}, NearVal::float(),  
FarVal::float()) -> {integer(), ObjX::float(), ObjY::float(), ObjZ::float(),  
ObjW::float()}
```

See **external** documentation.



---

## gl

Erlang module

Standard OpenGL api. See [www.opengl.org](http://www.opengl.org)

Booleans are represented by integers 0 and 1.

## DATA TYPES

`clamp()`

A float clamped between 0.0 - 1.0

`enum()`

An integer defined in gl.hrl

`offset()`

An integer which is an offset in an array

`wx_mem()`

see wx.erl on memory allocation functions

## Exports

**`accum(Op::enum(), Value::float()) -> ok`**

See **external** documentation.

**`alphaFunc(Func::enum(), Ref::clamp()) -> ok`**

See **external** documentation.

**`areTexturesResident(Textures::[integer()]) -> {0 | 1, Residences::[0 | 1]}`**

See **external** documentation.

**`arrayElement(I::integer()) -> ok`**

See **external** documentation.

**`begin(Mode::enum()) -> ok`**

See **external** documentation.

**`bindTexture(Target::enum(), Texture::integer()) -> ok`**

See **external** documentation.

**`bitmap(Width::integer(), Height::integer(), Xorig::float(), Yorig::float(), Xmove::float(), Ymove::float(), Bitmap::offset() | binary()) -> ok`**

See **external** documentation.

**blendFunc(Sfactor::enum(), Dfactor::enum()) -> ok**

See **external** documentation.

**callList(List::integer()) -> ok**

See **external** documentation.

**callLists(Lists::[integer()]) -> ok**

See **external** documentation.

**clear(Mask::integer()) -> ok**

See **external** documentation.

**clearAccum(Red::float(), Green::float(), Blue::float(), Alpha::float()) -> ok**

See **external** documentation.

**clearColor(Red::clamp(), Green::clamp(), Blue::clamp(), Alpha::clamp()) -> ok**

See **external** documentation.

**clearDepth(Depth::clamp()) -> ok**

See **external** documentation.

**clearIndex(C::float()) -> ok**

See **external** documentation.

**clearStencil(S::integer()) -> ok**

See **external** documentation.

**clipPlane(Plane::enum(), Equation::{float()}) -> ok**

See **external** documentation.

**color3b(Red::integer(), Green::integer(), Blue::integer()) -> ok**

See **external** documentation.

**color3bv(X1::{Red, Green, Blue}) -> ok**

Equivalent to *color3b(Red, Green, Blue)*.

**color3d(Red::float(), Green::float(), Blue::float()) -> ok**

See **external** documentation.

**color3dv(X1::{Red, Green, Blue}) -> ok**

Equivalent to *color3d(Red, Green, Blue)*.

---

```
color3f(Red::float(), Green::float(), Blue::float()) -> ok
```

See **external** documentation.

```
color3fv(X1::{Red, Green, Blue}) -> ok
```

Equivalent to *color3f(Red, Green, Blue)*.

```
color3i(Red::integer(), Green::integer(), Blue::integer()) -> ok
```

See **external** documentation.

```
color3iv(X1::{Red, Green, Blue}) -> ok
```

Equivalent to *color3i(Red, Green, Blue)*.

```
color3s(Red::integer(), Green::integer(), Blue::integer()) -> ok
```

See **external** documentation.

```
color3sv(X1::{Red, Green, Blue}) -> ok
```

Equivalent to *color3s(Red, Green, Blue)*.

```
color3ub(Red::integer(), Green::integer(), Blue::integer()) -> ok
```

See **external** documentation.

```
color3ubv(X1::{Red, Green, Blue}) -> ok
```

Equivalent to *color3ub(Red, Green, Blue)*.

```
color3ui(Red::integer(), Green::integer(), Blue::integer()) -> ok
```

See **external** documentation.

```
color3uiv(X1::{Red, Green, Blue}) -> ok
```

Equivalent to *color3ui(Red, Green, Blue)*.

```
color3us(Red::integer(), Green::integer(), Blue::integer()) -> ok
```

See **external** documentation.

```
color3usv(X1::{Red, Green, Blue}) -> ok
```

Equivalent to *color3us(Red, Green, Blue)*.

```
color4b(Red::integer(), Green::integer(), Blue::integer(), Alpha::integer())  
-> ok
```

See **external** documentation.

```
color4bv(X1::{Red, Green, Blue, Alpha}) -> ok
```

Equivalent to *color4b(Red, Green, Blue, Alpha)*.

`color4d(Red::float(), Green::float(), Blue::float(), Alpha::float()) -> ok`

See **external** documentation.

`color4dv(X1::{Red, Green, Blue, Alpha}) -> ok`

Equivalent to *color4d(Red, Green, Blue, Alpha)*.

`color4f(Red::float(), Green::float(), Blue::float(), Alpha::float()) -> ok`

See **external** documentation.

`color4fv(X1::{Red, Green, Blue, Alpha}) -> ok`

Equivalent to *color4f(Red, Green, Blue, Alpha)*.

`color4i(Red::integer(), Green::integer(), Blue::integer(), Alpha::integer())  
-> ok`

See **external** documentation.

`color4iv(X1::{Red, Green, Blue, Alpha}) -> ok`

Equivalent to *color4i(Red, Green, Blue, Alpha)*.

`color4s(Red::integer(), Green::integer(), Blue::integer(), Alpha::integer())  
-> ok`

See **external** documentation.

`color4sv(X1::{Red, Green, Blue, Alpha}) -> ok`

Equivalent to *color4s(Red, Green, Blue, Alpha)*.

`color4ub(Red::integer(), Green::integer(), Blue::integer(), Alpha::integer())  
-> ok`

See **external** documentation.

`color4ubv(X1::{Red, Green, Blue, Alpha}) -> ok`

Equivalent to *color4ub(Red, Green, Blue, Alpha)*.

`color4ui(Red::integer(), Green::integer(), Blue::integer(), Alpha::integer())  
-> ok`

See **external** documentation.

`color4uiv(X1::{Red, Green, Blue, Alpha}) -> ok`

Equivalent to *color4ui(Red, Green, Blue, Alpha)*.

`color4us(Red::integer(), Green::integer(), Blue::integer(), Alpha::integer())  
-> ok`

See **external** documentation.

---

```
color4usv(X1::{Red, Green, Blue, Alpha}) -> ok
```

Equivalent to *color4us(Red, Green, Blue, Alpha)*.

```
colorMask(Red::0 | 1, Green::0 | 1, Blue::0 | 1, Alpha::0 | 1) -> ok
```

See **external** documentation.

```
colorMaterial(Face::enum(), Mode::enum()) -> ok
```

See **external** documentation.

```
colorPointer(Size::integer(), Type::enum(), Stride::integer(),  
Pointer::offset() | binary()) -> ok
```

See **external** documentation.

```
copyPixels(X::integer(), Y::integer(), Width::integer(), Height::integer(),  
Type::enum()) -> ok
```

See **external** documentation.

```
copyTexImage1D(Target::enum(), Level::integer(), InternalFormat::enum(),  
X::integer(), Y::integer(), Width::integer(), Border::integer()) -> ok
```

See **external** documentation.

```
copyTexImage2D(Target::enum(), Level::integer(), InternalFormat::enum(),  
X::integer(), Y::integer(), Width::integer(), Height::integer(),  
Border::integer()) -> ok
```

See **external** documentation.

```
copyTexSubImage1D(Target::enum(), Level::integer(), Xoffset::integer(),  
X::integer(), Y::integer(), Width::integer()) -> ok
```

See **external** documentation.

```
copyTexSubImage2D(Target::enum(), Level::integer(), Xoffset::integer(),  
Yoffset::integer(), X::integer(), Y::integer(), Width::integer(),  
Height::integer()) -> ok
```

See **external** documentation.

```
cullFace(Mode::enum()) -> ok
```

See **external** documentation.

```
deleteLists(List::integer(), Range::integer()) -> ok
```

See **external** documentation.

```
deleteTextures(Textures::[integer()]) -> ok
```

See **external** documentation.

**depthFunc(Func::enum()) -> ok**

See **external** documentation.

**depthMask(Flag::0 | 1) -> ok**

See **external** documentation.

**depthRange(ZNear::clamp(), ZFar::clamp()) -> ok**

See **external** documentation.

**disable(Cap::enum()) -> ok**

See **external** documentation.

**disableClientState(Array::enum()) -> ok**

See **external** documentation.

**drawArrays(Mode::enum(), First::integer(), Count::integer()) -> ok**

See **external** documentation.

**drawBuffer(Mode::enum()) -> ok**

See **external** documentation.

**drawElements(Mode::enum(), Count::integer(), Type::enum(), Indices::offset()  
| binary()) -> ok**

See **external** documentation.

**drawPixels(Width::integer(), Height::integer(), Format::enum(), Type::enum(),  
Pixels::offset() | binary()) -> ok**

See **external** documentation.

**edgeFlag(Flag::0 | 1) -> ok**

See **external** documentation.

**edgeFlagPointer(Stride::integer(), Pointer::offset() | binary()) -> ok**

See **external** documentation.

**edgeFlagv(X1::{Flag}) -> ok**

Equivalent to *edgeFlag(Flag)*.

**enable(Cap::enum()) -> ok**

See **external** documentation.

**enableClientState(Array::enum()) -> ok**

See **external** documentation.

**end()** -> ok

See **external** documentation.

**endList()** -> ok

See **external** documentation.

**evalCoord1d(U::float())** -> ok

See **external** documentation.

**evalCoord1dv(X1::{U})** -> ok

Equivalent to *evalCoord1d(U)*.

**evalCoord1f(U::float())** -> ok

See **external** documentation.

**evalCoord1fv(X1::{U})** -> ok

Equivalent to *evalCoord1f(U)*.

**evalCoord2d(U::float(), V::float())** -> ok

See **external** documentation.

**evalCoord2dv(X1::{U, V})** -> ok

Equivalent to *evalCoord2d(U, V)*.

**evalCoord2f(U::float(), V::float())** -> ok

See **external** documentation.

**evalCoord2fv(X1::{U, V})** -> ok

Equivalent to *evalCoord2f(U, V)*.

**evalMesh1(Mode::enum(), I1::integer(), I2::integer())** -> ok

See **external** documentation.

**evalMesh2(Mode::enum(), I1::integer(), I2::integer(), J1::integer(), J2::integer())** -> ok

See **external** documentation.

**evalPoint1(I::integer())** -> ok

See **external** documentation.

**evalPoint2(I::integer(), J::integer())** -> ok

See **external** documentation.

**feedbackBuffer(Size::integer(), Type::enum(), Buffer::wx\_mem())** (see module **wx**) -> **ok**

See **external** documentation.

**finish()** -> **ok**

See **external** documentation.

**flush()** -> **ok**

See **external** documentation.

**fogf(Pname::enum(), Param::float())** -> **ok**

See **external** documentation.

**fogfv(Pname::enum(), Params::{float()})** -> **ok**

See **external** documentation.

**fogi(Pname::enum(), Param::integer())** -> **ok**

See **external** documentation.

**fogiv(Pname::enum(), Params::{integer()})** -> **ok**

See **external** documentation.

**frontFace(Mode::enum())** -> **ok**

See **external** documentation.

**frustum(Left::float(), Right::float(), Bottom::float(), Top::float(), ZNear::float(), ZFar::float())** -> **ok**

See **external** documentation.

**genLists(Range::integer())** -> **integer()**

See **external** documentation.

**genTextures(N::integer())** -> [**integer()**]

See **external** documentation.

**getBooleanv(Pname::enum())** -> [0 | 1]

See **external** documentation.

**getClipPlane(Plane::enum())** -> {**float()**}

See **external** documentation.

**getDoublev(Pname::enum())** -> [**float()**]

See **external** documentation.



`getError()` -> `enum()`

See **external** documentation.

`getFloatv(Pname::enum())` -> `[float()]`

See **external** documentation.

`getInterv(Pname::enum())` -> `[integer()]`

See **external** documentation.

`getLightfv(Light::enum(), Pname::enum())` -> `{float()}`

See **external** documentation.

`getLightiv(Light::enum(), Pname::enum())` -> `{integer()}`

See **external** documentation.

`getMapdv(Target::enum(), Query::enum(), V::wx_mem() (see module wx))` -> `ok`

See **external** documentation.

`getMapfv(Target::enum(), Query::enum(), V::wx_mem() (see module wx))` -> `ok`

See **external** documentation.

`getMapiv(Target::enum(), Query::enum(), V::wx_mem() (see module wx))` -> `ok`

See **external** documentation.

`getMaterialfv(Face::enum(), Pname::enum())` -> `{float()}`

See **external** documentation.

`getMaterialiv(Face::enum(), Pname::enum())` -> `{integer()}`

See **external** documentation.

`getPixelMapfv(Map::enum(), Values::wx_mem() (see module wx))` -> `ok`

See **external** documentation.

`getPixelMapuiv(Map::enum(), Values::wx_mem() (see module wx))` -> `ok`

See **external** documentation.

`getPixelMapusv(Map::enum(), Values::wx_mem() (see module wx))` -> `ok`

See **external** documentation.

`getPolygonStipple()` -> `binary()`

See **external** documentation.

`getString(Name::enum()) -> string()`

See **external** documentation.

`getTexEnvfv(Target::enum(), Pname::enum()) -> {float()}`

See **external** documentation.

`getTexEnviv(Target::enum(), Pname::enum()) -> {integer()}`

See **external** documentation.

`getTexGendv(Coord::enum(), Pname::enum()) -> {float()}`

See **external** documentation.

`getTexGenfv(Coord::enum(), Pname::enum()) -> {float()}`

See **external** documentation.

`getTexGeniv(Coord::enum(), Pname::enum()) -> {integer()}`

See **external** documentation.

`getTexImage(Target::enum(), Level::integer(), Format::enum(), Type::enum(),  
Pixels::wx_mem() (see module wx)) -> ok`

See **external** documentation.

`getTexLevelParameterfv(Target::enum(), Level::integer(), Pname::enum()) ->  
{float()}`

See **external** documentation.

`getTexLevelParameteriv(Target::enum(), Level::integer(), Pname::enum()) ->  
{integer()}`

See **external** documentation.

`getTexParameterfv(Target::enum(), Pname::enum()) -> {float()}`

See **external** documentation.

`getTexParameteriv(Target::enum(), Pname::enum()) -> {integer()}`

See **external** documentation.

`hint(Target::enum(), Mode::enum()) -> ok`

See **external** documentation.

`indexMask(Mask::integer()) -> ok`

See **external** documentation.

**indexPointer**(Type::enum(), Stride::integer(), Pointer::offset() | binary()) -> ok

See **external** documentation.

**indexd**(C::float()) -> ok

See **external** documentation.

**indexdv**(X1::{C}) -> ok

Equivalent to *indexd(C)*.

**indexf**(C::float()) -> ok

See **external** documentation.

**indexfv**(X1::{C}) -> ok

Equivalent to *indexf(C)*.

**indexi**(C::integer()) -> ok

See **external** documentation.

**indexiv**(X1::{C}) -> ok

Equivalent to *indexi(C)*.

**indexs**(C::integer()) -> ok

See **external** documentation.

**indexsv**(X1::{C}) -> ok

Equivalent to *indexs(C)*.

**indexub**(C::integer()) -> ok

See **external** documentation.

**indexubv**(X1::{C}) -> ok

Equivalent to *indexub(C)*.

**initNames**() -> ok

See **external** documentation.

**interleavedArrays**(Format::enum(), Stride::integer(), Pointer::offset() | binary()) -> ok

See **external** documentation.

**isEnabled**(Cap::enum()) -> 0 | 1

See **external** documentation.

`isList(List::integer()) -> 0 | 1`

See **external** documentation.

`isTexture(Texture::integer()) -> 0 | 1`

See **external** documentation.

`lightModelf(Pname::enum(), Param::float()) -> ok`

See **external** documentation.

`lightModelfv(Pname::enum(), Params::{float()}) -> ok`

See **external** documentation.

`lightModeli(Pname::enum(), Param::integer()) -> ok`

See **external** documentation.

`lightModeliv(Pname::enum(), Params::{integer()}) -> ok`

See **external** documentation.

`lightf(Light::enum(), Pname::enum(), Param::float()) -> ok`

See **external** documentation.

`lightfv(Light::enum(), Pname::enum(), Params::{float()}) -> ok`

See **external** documentation.

`lighti(Light::enum(), Pname::enum(), Param::integer()) -> ok`

See **external** documentation.

`lightiv(Light::enum(), Pname::enum(), Params::{integer()}) -> ok`

See **external** documentation.

`lineStipple(Factor::integer(), Pattern::integer()) -> ok`

See **external** documentation.

`lineWidth(Width::float()) -> ok`

See **external** documentation.

`listBase(Base::integer()) -> ok`

See **external** documentation.

`loadIdentity() -> ok`

See **external** documentation.

`loadMatrixd(M::{float()}) -> ok`

See **external** documentation.

`loadMatrixf(M::{float()}) -> ok`

See **external** documentation.

`loadName(Name::integer()) -> ok`

See **external** documentation.

`logicOp(Opcode::enum()) -> ok`

See **external** documentation.

`map1d(Target::enum(), U1::float(), U2::float(), Stride::integer(),  
Order::integer(), Points::binary()) -> ok`

See **external** documentation.

`map1f(Target::enum(), U1::float(), U2::float(), Stride::integer(),  
Order::integer(), Points::binary()) -> ok`

See **external** documentation.

`map2d(Target::enum(), U1::float(), U2::float(), Ustride::integer(),  
Uorder::integer(), V1::float(), V2::float(), Vstride::integer(),  
Vorder::integer(), Points::binary()) -> ok`

See **external** documentation.

`map2f(Target::enum(), U1::float(), U2::float(), Ustride::integer(),  
Uorder::integer(), V1::float(), V2::float(), Vstride::integer(),  
Vorder::integer(), Points::binary()) -> ok`

See **external** documentation.

`mapGrid1d(Un::integer(), U1::float(), U2::float()) -> ok`

See **external** documentation.

`mapGrid1f(Un::integer(), U1::float(), U2::float()) -> ok`

See **external** documentation.

`mapGrid2d(Un::integer(), U1::float(), U2::float(), Vn::integer(),  
V1::float(), V2::float()) -> ok`

See **external** documentation.

`mapGrid2f(Un::integer(), U1::float(), U2::float(), Vn::integer(),  
V1::float(), V2::float()) -> ok`

See **external** documentation.

**materialf**(Face::enum(), Pname::enum(), Param::float()) -> ok

See **external** documentation.

**materialfv**(Face::enum(), Pname::enum(), Params::{float()}) -> ok

See **external** documentation.

**materiali**(Face::enum(), Pname::enum(), Param::integer()) -> ok

See **external** documentation.

**materialiv**(Face::enum(), Pname::enum(), Params::{integer()}) -> ok

See **external** documentation.

**matrixMode**(Mode::enum()) -> ok

See **external** documentation.

**multMatrixd**(M::{float()}) -> ok

See **external** documentation.

**multMatrixf**(M::{float()}) -> ok

See **external** documentation.

**newList**(List::integer(), Mode::enum()) -> ok

See **external** documentation.

**normal3b**(Nx::integer(), Ny::integer(), Nz::integer()) -> ok

See **external** documentation.

**normal3bv**(Xl::{Nx, Ny, Nz}) -> ok

Equivalent to *normal3b*(Nx, Ny, Nz).

**normal3d**(Nx::float(), Ny::float(), Nz::float()) -> ok

See **external** documentation.

**normal3dv**(Xl::{Nx, Ny, Nz}) -> ok

Equivalent to *normal3d*(Nx, Ny, Nz).

**normal3f**(Nx::float(), Ny::float(), Nz::float()) -> ok

See **external** documentation.

**normal3fv**(Xl::{Nx, Ny, Nz}) -> ok

Equivalent to *normal3f*(Nx, Ny, Nz).

---

```
normal3i(Nx::integer(), Ny::integer(), Nz::integer()) -> ok
```

See **external** documentation.

```
normal3iv(X1::{Nx, Ny, Nz}) -> ok
```

Equivalent to *normal3i(Nx, Ny, Nz)*.

```
normal3s(Nx::integer(), Ny::integer(), Nz::integer()) -> ok
```

See **external** documentation.

```
normal3sv(X1::{Nx, Ny, Nz}) -> ok
```

Equivalent to *normal3s(Nx, Ny, Nz)*.

```
normalPointer(Type::enum(), Stride::integer(), Pointer::offset() | binary())  
-> ok
```

See **external** documentation.

```
ortho(Left::float(), Right::float(), Bottom::float(), Top::float(),  
ZNear::float(), ZFar::float()) -> ok
```

See **external** documentation.

```
passThrough(Token::float()) -> ok
```

See **external** documentation.

```
pixelMapfv(Map::enum(), Mapsize::integer(), Values::binary()) -> ok
```

See **external** documentation.

```
pixelMapuiv(Map::enum(), Mapsize::integer(), Values::binary()) -> ok
```

See **external** documentation.

```
pixelMapusv(Map::enum(), Mapsize::integer(), Values::binary()) -> ok
```

See **external** documentation.

```
pixelStoref(Pname::enum(), Param::float()) -> ok
```

See **external** documentation.

```
pixelStorei(Pname::enum(), Param::integer()) -> ok
```

See **external** documentation.

```
pixelTransferf(Pname::enum(), Param::float()) -> ok
```

See **external** documentation.

```
pixelTransferi(Pname::enum(), Param::integer()) -> ok
```

See **external** documentation.

**pixelZoom(Xfactor::float(), Yfactor::float()) -> ok**

See **external** documentation.

**pointSize(Size::float()) -> ok**

See **external** documentation.

**polygonMode(Face::enum(), Mode::enum()) -> ok**

See **external** documentation.

**polygonOffset(Factor::float(), Units::float()) -> ok**

See **external** documentation.

**polygonStipple(Mask::binary()) -> ok**

See **external** documentation.

**popAttrib() -> ok**

See **external** documentation.

**popClientAttrib() -> ok**

See **external** documentation.

**popMatrix() -> ok**

See **external** documentation.

**popName() -> ok**

See **external** documentation.

**prioritizeTextures(Textures::[integer()], Priorities::[clamp()]) -> ok**

See **external** documentation.

**pushAttrib(Mask::integer()) -> ok**

See **external** documentation.

**pushClientAttrib(Mask::integer()) -> ok**

See **external** documentation.

**pushMatrix() -> ok**

See **external** documentation.

**pushName(Name::integer()) -> ok**

See **external** documentation.



---

```
rasterPos2d(X::float(), Y::float()) -> ok
```

See **external** documentation.

```
rasterPos2dv(X1::{X, Y}) -> ok
```

Equivalent to *rasterPos2d(X, Y)*.

```
rasterPos2f(X::float(), Y::float()) -> ok
```

See **external** documentation.

```
rasterPos2fv(X1::{X, Y}) -> ok
```

Equivalent to *rasterPos2f(X, Y)*.

```
rasterPos2i(X::integer(), Y::integer()) -> ok
```

See **external** documentation.

```
rasterPos2iv(X1::{X, Y}) -> ok
```

Equivalent to *rasterPos2i(X, Y)*.

```
rasterPos2s(X::integer(), Y::integer()) -> ok
```

See **external** documentation.

```
rasterPos2sv(X1::{X, Y}) -> ok
```

Equivalent to *rasterPos2s(X, Y)*.

```
rasterPos3d(X::float(), Y::float(), Z::float()) -> ok
```

See **external** documentation.

```
rasterPos3dv(X1::{X, Y, Z}) -> ok
```

Equivalent to *rasterPos3d(X, Y, Z)*.

```
rasterPos3f(X::float(), Y::float(), Z::float()) -> ok
```

See **external** documentation.

```
rasterPos3fv(X1::{X, Y, Z}) -> ok
```

Equivalent to *rasterPos3f(X, Y, Z)*.

```
rasterPos3i(X::integer(), Y::integer(), Z::integer()) -> ok
```

See **external** documentation.

```
rasterPos3iv(X1::{X, Y, Z}) -> ok
```

Equivalent to *rasterPos3i(X, Y, Z)*.

**rasterPos3s**(X::integer(), Y::integer(), Z::integer()) -> ok

See **external** documentation.

**rasterPos3sv**(X1::{X, Y, Z}) -> ok

Equivalent to *rasterPos3s*(X, Y, Z).

**rasterPos4d**(X::float(), Y::float(), Z::float(), W::float()) -> ok

See **external** documentation.

**rasterPos4dv**(X1::{X, Y, Z, W}) -> ok

Equivalent to *rasterPos4d*(X, Y, Z, W).

**rasterPos4f**(X::float(), Y::float(), Z::float(), W::float()) -> ok

See **external** documentation.

**rasterPos4fv**(X1::{X, Y, Z, W}) -> ok

Equivalent to *rasterPos4f*(X, Y, Z, W).

**rasterPos4i**(X::integer(), Y::integer(), Z::integer(), W::integer()) -> ok

See **external** documentation.

**rasterPos4iv**(X1::{X, Y, Z, W}) -> ok

Equivalent to *rasterPos4i*(X, Y, Z, W).

**rasterPos4s**(X::integer(), Y::integer(), Z::integer(), W::integer()) -> ok

See **external** documentation.

**rasterPos4sv**(X1::{X, Y, Z, W}) -> ok

Equivalent to *rasterPos4s*(X, Y, Z, W).

**readBuffer**(Mode::enum()) -> ok

See **external** documentation.

**readPixels**(X::integer(), Y::integer(), Width::integer(), Height::integer(),  
Format::enum(), Type::enum(), Pixels::wx\_mem() (see module wx)) -> ok

See **external** documentation.

**rectd**(X1::float(), Y1::float(), X2::float(), Y2::float()) -> ok

See **external** documentation.

**rectdv**(V1::{float()}, V2::{float()}) -> ok

See **external** documentation.

**rectf**(X1::float(), Y1::float(), X2::float(), Y2::float()) -> ok

See **external** documentation.

**rectfv**(V1::{float()}, V2::{float()}) -> ok

See **external** documentation.

**recti**(X1::integer(), Y1::integer(), X2::integer(), Y2::integer()) -> ok

See **external** documentation.

**rectiv**(V1::{integer()}, V2::{integer()}) -> ok

See **external** documentation.

**rects**(X1::integer(), Y1::integer(), X2::integer(), Y2::integer()) -> ok

See **external** documentation.

**rectsv**(V1::{integer()}, V2::{integer()}) -> ok

See **external** documentation.

**renderMode**(Mode::enum()) -> integer()

See **external** documentation.

**rotated**(Angle::float(), X::float(), Y::float(), Z::float()) -> ok

See **external** documentation.

**rotatef**(Angle::float(), X::float(), Y::float(), Z::float()) -> ok

See **external** documentation.

**scaled**(X::float(), Y::float(), Z::float()) -> ok

See **external** documentation.

**scalef**(X::float(), Y::float(), Z::float()) -> ok

See **external** documentation.

**scissor**(X::integer(), Y::integer(), Width::integer(), Height::integer()) -> ok

See **external** documentation.

**selectBuffer**(Size::integer(), Buffer::wx\_mem() (see module wx)) -> ok

See **external** documentation.

**shadeModel**(Mode::enum()) -> ok

See **external** documentation.

**stencilFunc(Func::enum(), Ref::integer(), Mask::integer()) -> ok**

See **external** documentation.

**stencilMask(Mask::integer()) -> ok**

See **external** documentation.

**stencilOp(Fail::enum(), Zfail::enum(), Zpass::enum()) -> ok**

See **external** documentation.

**texCoord1d(S::float()) -> ok**

See **external** documentation.

**texCoord1dv(X1::{S}) -> ok**

Equivalent to *texCoord1d(S)*.

**texCoord1f(S::float()) -> ok**

See **external** documentation.

**texCoord1fv(X1::{S}) -> ok**

Equivalent to *texCoord1f(S)*.

**texCoord1i(S::integer()) -> ok**

See **external** documentation.

**texCoord1iv(X1::{S}) -> ok**

Equivalent to *texCoord1i(S)*.

**texCoord1s(S::integer()) -> ok**

See **external** documentation.

**texCoord1sv(X1::{S}) -> ok**

Equivalent to *texCoord1s(S)*.

**texCoord2d(S::float(), T::float()) -> ok**

See **external** documentation.

**texCoord2dv(X1::{S, T}) -> ok**

Equivalent to *texCoord2d(S, T)*.

**texCoord2f(S::float(), T::float()) -> ok**

See **external** documentation.

`texCoord2fv(X1::{S, T}) -> ok`

Equivalent to *texCoord2f(S, T)*.

`texCoord2i(S::integer(), T::integer()) -> ok`

See **external** documentation.

`texCoord2iv(X1::{S, T}) -> ok`

Equivalent to *texCoord2i(S, T)*.

`texCoord2s(S::integer(), T::integer()) -> ok`

See **external** documentation.

`texCoord2sv(X1::{S, T}) -> ok`

Equivalent to *texCoord2s(S, T)*.

`texCoord3d(S::float(), T::float(), R::float()) -> ok`

See **external** documentation.

`texCoord3dv(X1::{S, T, R}) -> ok`

Equivalent to *texCoord3d(S, T, R)*.

`texCoord3f(S::float(), T::float(), R::float()) -> ok`

See **external** documentation.

`texCoord3fv(X1::{S, T, R}) -> ok`

Equivalent to *texCoord3f(S, T, R)*.

`texCoord3i(S::integer(), T::integer(), R::integer()) -> ok`

See **external** documentation.

`texCoord3iv(X1::{S, T, R}) -> ok`

Equivalent to *texCoord3i(S, T, R)*.

`texCoord3s(S::integer(), T::integer(), R::integer()) -> ok`

See **external** documentation.

`texCoord3sv(X1::{S, T, R}) -> ok`

Equivalent to *texCoord3s(S, T, R)*.

`texCoord4d(S::float(), T::float(), R::float(), Q::float()) -> ok`

See **external** documentation.

```
texCoord4dv(Xl::{S, T, R, Q}) -> ok
```

Equivalent to *texCoord4d(S, T, R, Q)*.

```
texCoord4f(S::float(), T::float(), R::float(), Q::float()) -> ok
```

See **external** documentation.

```
texCoord4fv(Xl::{S, T, R, Q}) -> ok
```

Equivalent to *texCoord4f(S, T, R, Q)*.

```
texCoord4i(S::integer(), T::integer(), R::integer(), Q::integer()) -> ok
```

See **external** documentation.

```
texCoord4iv(Xl::{S, T, R, Q}) -> ok
```

Equivalent to *texCoord4i(S, T, R, Q)*.

```
texCoord4s(S::integer(), T::integer(), R::integer(), Q::integer()) -> ok
```

See **external** documentation.

```
texCoord4sv(Xl::{S, T, R, Q}) -> ok
```

Equivalent to *texCoord4s(S, T, R, Q)*.

```
texCoordPointer(Size::integer(), Type::enum(), Stride::integer(),  
Pointer::offset() | binary()) -> ok
```

See **external** documentation.

```
texEnvf(Target::enum(), Pname::enum(), Param::float()) -> ok
```

See **external** documentation.

```
texEnvfv(Target::enum(), Pname::enum(), Params::{float()}) -> ok
```

See **external** documentation.

```
texEnvi(Target::enum(), Pname::enum(), Param::integer()) -> ok
```

See **external** documentation.

```
texEnviv(Target::enum(), Pname::enum(), Params::{integer()}) -> ok
```

See **external** documentation.

```
texGend(Coord::enum(), Pname::enum(), Param::float()) -> ok
```

See **external** documentation.

```
texGendv(Coord::enum(), Pname::enum(), Params::{float()}) -> ok
```

See **external** documentation.

---

```
texGenf(Coord::enum(), Pname::enum(), Param::float()) -> ok
```

See **external** documentation.

```
texGenfv(Coord::enum(), Pname::enum(), Params::{float()}) -> ok
```

See **external** documentation.

```
texGeni(Coord::enum(), Pname::enum(), Param::integer()) -> ok
```

See **external** documentation.

```
texGeniv(Coord::enum(), Pname::enum(), Params::{integer()}) -> ok
```

See **external** documentation.

```
texImage1D(Target::enum(), Level::integer(), Internalformat::integer(),  
Width::integer(), Border::integer(), Format::enum(), Type::enum(),  
Pixels::offset() | binary()) -> ok
```

See **external** documentation.

```
texImage2D(Target::enum(), Level::integer(), Internalformat::integer(),  
Width::integer(), Height::integer(), Border::integer(), Format::enum(),  
Type::enum(), Pixels::offset() | binary()) -> ok
```

See **external** documentation.

```
texParameterf(Target::enum(), Pname::enum(), Param::float()) -> ok
```

See **external** documentation.

```
texParameterfv(Target::enum(), Pname::enum(), Params::{float()}) -> ok
```

See **external** documentation.

```
texParameteri(Target::enum(), Pname::enum(), Param::integer()) -> ok
```

See **external** documentation.

```
texParameteriv(Target::enum(), Pname::enum(), Params::{integer()}) -> ok
```

See **external** documentation.

```
texSubImage1D(Target::enum(), Level::integer(), Xoffset::integer(),  
Width::integer(), Format::enum(), Type::enum(), Pixels::offset() | binary())  
-> ok
```

See **external** documentation.

```
texSubImage2D(Target::enum(), Level::integer(), Xoffset::integer(),  
Yoffset::integer(), Width::integer(), Height::integer(), Format::enum(),  
Type::enum(), Pixels::offset() | binary()) -> ok
```

See **external** documentation.

**translated(X::float(), Y::float(), Z::float()) -> ok**

See **external** documentation.

**translatef(X::float(), Y::float(), Z::float()) -> ok**

See **external** documentation.

**vertex2d(X::float(), Y::float()) -> ok**

See **external** documentation.

**vertex2dv(X1::{X, Y}) -> ok**

Equivalent to *vertex2d(X, Y)*.

**vertex2f(X::float(), Y::float()) -> ok**

See **external** documentation.

**vertex2fv(X1::{X, Y}) -> ok**

Equivalent to *vertex2f(X, Y)*.

**vertex2i(X::integer(), Y::integer()) -> ok**

See **external** documentation.

**vertex2iv(X1::{X, Y}) -> ok**

Equivalent to *vertex2i(X, Y)*.

**vertex2s(X::integer(), Y::integer()) -> ok**

See **external** documentation.

**vertex2sv(X1::{X, Y}) -> ok**

Equivalent to *vertex2s(X, Y)*.

**vertex3d(X::float(), Y::float(), Z::float()) -> ok**

See **external** documentation.

**vertex3dv(X1::{X, Y, Z}) -> ok**

Equivalent to *vertex3d(X, Y, Z)*.

**vertex3f(X::float(), Y::float(), Z::float()) -> ok**

See **external** documentation.

**vertex3fv(X1::{X, Y, Z}) -> ok**

Equivalent to *vertex3f(X, Y, Z)*.



**vertex3i**(X::integer(), Y::integer(), Z::integer()) -> ok

See **external** documentation.

**vertex3iv**(X1::{X, Y, Z}) -> ok

Equivalent to *vertex3i*(X, Y, Z).

**vertex3s**(X::integer(), Y::integer(), Z::integer()) -> ok

See **external** documentation.

**vertex3sv**(X1::{X, Y, Z}) -> ok

Equivalent to *vertex3s*(X, Y, Z).

**vertex4d**(X::float(), Y::float(), Z::float(), W::float()) -> ok

See **external** documentation.

**vertex4dv**(X1::{X, Y, Z, W}) -> ok

Equivalent to *vertex4d*(X, Y, Z, W).

**vertex4f**(X::float(), Y::float(), Z::float(), W::float()) -> ok

See **external** documentation.

**vertex4fv**(X1::{X, Y, Z, W}) -> ok

Equivalent to *vertex4f*(X, Y, Z, W).

**vertex4i**(X::integer(), Y::integer(), Z::integer(), W::integer()) -> ok

See **external** documentation.

**vertex4iv**(X1::{X, Y, Z, W}) -> ok

Equivalent to *vertex4i*(X, Y, Z, W).

**vertex4s**(X::integer(), Y::integer(), Z::integer(), W::integer()) -> ok

See **external** documentation.

**vertex4sv**(X1::{X, Y, Z, W}) -> ok

Equivalent to *vertex4s*(X, Y, Z, W).

**vertexPointer**(Size::integer(), Type::enum(), Stride::integer(),  
Pointer::offset() | binary()) -> ok

See **external** documentation.

**viewport**(X::integer(), Y::integer(), Width::integer(), Height::integer()) ->  
ok

See **external** documentation.

`blendColor(Red::clamp(), Green::clamp(), Blue::clamp(), Alpha::clamp()) -> ok`

See **external** documentation.

`blendEquation(Mode::enum()) -> ok`

See **external** documentation.

`drawRangeElements(Mode::enum(), Start::integer(), End::integer(),  
Count::integer(), Type::enum(), Indices::offset() | binary()) -> ok`

See **external** documentation.

`texImage3D(Target::enum(), Level::integer(), Internalformat::integer(),  
Width::integer(), Height::integer(), Depth::integer(), Border::integer(),  
Format::enum(), Type::enum(), Pixels::offset() | binary()) -> ok`

See **external** documentation.

`texSubImage3D(Target::enum(), Level::integer(), Xoffset::integer(),  
Yoffset::integer(), Zoffset::integer(), Width::integer(), Height::integer(),  
Depth::integer(), Format::enum(), Type::enum(), Pixels::offset() | binary())  
-> ok`

See **external** documentation.

`copyTexSubImage3D(Target::enum(), Level::integer(), Xoffset::integer(),  
Yoffset::integer(), Zoffset::integer(), X::integer(), Y::integer(),  
Width::integer(), Height::integer()) -> ok`

See **external** documentation.

`colorTable(Target::enum(), Internalformat::enum(), Width::integer(),  
Format::enum(), Type::enum(), Table::offset() | binary()) -> ok`

See **external** documentation.

`colorTableParameterfv(Target::enum(), Pname::enum(), Params::{float()}) -> ok`

See **external** documentation.

`colorTableParameteriv(Target::enum(), Pname::enum(), Params::{integer()}) ->  
ok`

See **external** documentation.

`copyColorTable(Target::enum(), Internalformat::enum(), X::integer(),  
Y::integer(), Width::integer()) -> ok`

See **external** documentation.

`getColorTable(Target::enum(), Format::enum(), Type::enum(), Table::wx_mem()  
(see module wx)) -> ok`

See **external** documentation.

---

```
getColorTableParameterfv(Target::enum(), Pname::enum()) -> {float()}
```

See **external** documentation.

```
getColorTableParameteriv(Target::enum(), Pname::enum()) -> {integer()}
```

See **external** documentation.

```
colorSubTable(Target::enum(), Start::integer(), Count::integer(),
Format::enum(), Type::enum(), Data::offset() | binary()) -> ok
```

See **external** documentation.

```
copyColorSubTable(Target::enum(), Start::integer(), X::integer(),
Y::integer(), Width::integer()) -> ok
```

See **external** documentation.

```
convolutionFilter1D(Target::enum(), Internalformat::enum(), Width::integer(),
Format::enum(), Type::enum(), Image::offset() | binary()) -> ok
```

See **external** documentation.

```
convolutionFilter2D(Target::enum(), Internalformat::enum(), Width::integer(),
Height::integer(), Format::enum(), Type::enum(), Image::offset() | binary())
-> ok
```

See **external** documentation.

```
convolutionParameterf(Target::enum(), Pname::enum(), Params::{float()}) -> ok
```

See **external** documentation.

```
convolutionParameterfv(Target, Pname, X3::{Params}) -> ok
```

Equivalent to *convolutionParameterf(Target, Pname, Params)*.

```
convolutionParameteri(Target::enum(), Pname::enum(), Params::{integer()}) ->
ok
```

See **external** documentation.

```
convolutionParameteriv(Target, Pname, X3::{Params}) -> ok
```

Equivalent to *convolutionParameteri(Target, Pname, Params)*.

```
copyConvolutionFilter1D(Target::enum(), Internalformat::enum(), X::integer(),
Y::integer(), Width::integer()) -> ok
```

See **external** documentation.

```
copyConvolutionFilter2D(Target::enum(), Internalformat::enum(), X::integer(),
Y::integer(), Width::integer(), Height::integer()) -> ok
```

See **external** documentation.

`getConvolutionFilter(Target::enum(), Format::enum(), Type::enum(),  
Image::wx_mem() (see module wx)) -> ok`

See **external** documentation.

`getConvolutionParameterfv(Target::enum(), Pname::enum()) -> {float()}`

See **external** documentation.

`getConvolutionParameteriv(Target::enum(), Pname::enum()) -> {integer()}`

See **external** documentation.

`separableFilter2D(Target::enum(), Internalformat::enum(), Width::integer(),  
Height::integer(), Format::enum(), Type::enum(), Row::offset() | binary(),  
Column::offset() | binary()) -> ok`

See **external** documentation.

`getHistogram(Target::enum(), Reset::0 | 1, Format::enum(), Type::enum(),  
Values::wx_mem() (see module wx)) -> ok`

See **external** documentation.

`getHistogramParameterfv(Target::enum(), Pname::enum()) -> {float()}`

See **external** documentation.

`getHistogramParameteriv(Target::enum(), Pname::enum()) -> {integer()}`

See **external** documentation.

`getMinmax(Target::enum(), Reset::0 | 1, Format::enum(), Type::enum(),  
Values::wx_mem() (see module wx)) -> ok`

See **external** documentation.

`getMinmaxParameterfv(Target::enum(), Pname::enum()) -> {float()}`

See **external** documentation.

`getMinmaxParameteriv(Target::enum(), Pname::enum()) -> {integer()}`

See **external** documentation.

`histogram(Target::enum(), Width::integer(), Internalformat::enum(), Sink::0 |  
1) -> ok`

See **external** documentation.

`minmax(Target::enum(), Internalformat::enum(), Sink::0 | 1) -> ok`

See **external** documentation.

`resetHistogram(Target::enum()) -> ok`

See **external** documentation.

---

```
resetMinmax(Target::enum()) -> ok
```

See **external** documentation.

```
activeTexture(Texture::enum()) -> ok
```

See **external** documentation.

```
sampleCoverage(Value::clamp(), Invert::0 | 1) -> ok
```

See **external** documentation.

```
compressedTexImage3D(Target::enum(), Level::integer(),  
Internalformat::enum(), Width::integer(), Height::integer(),  
Depth::integer(), Border::integer(), ImageSize::integer(), Data::offset() |  
binary()) -> ok
```

See **external** documentation.

```
compressedTexImage2D(Target::enum(), Level::integer(),  
Internalformat::enum(), Width::integer(), Height::integer(),  
Border::integer(), ImageSize::integer(), Data::offset() | binary()) -> ok
```

See **external** documentation.

```
compressedTexImage1D(Target::enum(), Level::integer(),  
Internalformat::enum(), Width::integer(), Border::integer(),  
ImageSize::integer(), Data::offset() | binary()) -> ok
```

See **external** documentation.

```
compressedTexSubImage3D(Target::enum(), Level::integer(), Xoffset::integer(),  
Yoffset::integer(), Zoffset::integer(), Width::integer(), Height::integer(),  
Depth::integer(), Format::enum(), ImageSize::integer(), Data::offset() |  
binary()) -> ok
```

See **external** documentation.

```
compressedTexSubImage2D(Target::enum(), Level::integer(), Xoffset::integer(),  
Yoffset::integer(), Width::integer(), Height::integer(), Format::enum(),  
ImageSize::integer(), Data::offset() | binary()) -> ok
```

See **external** documentation.

```
compressedTexSubImage1D(Target::enum(), Level::integer(), Xoffset::integer(),  
Width::integer(), Format::enum(), ImageSize::integer(), Data::offset() |  
binary()) -> ok
```

See **external** documentation.

```
getCompressedTexImage(Target::enum(), Level::integer(), Img::wx_mem() (see  
module wx)) -> ok
```

See **external** documentation.

`clientActiveTexture(Texture::enum()) -> ok`

See **external** documentation.

`multiTexCoord1d(Target::enum(), S::float()) -> ok`

See **external** documentation.

`multiTexCoord1dv(Target, X2::{S}) -> ok`

Equivalent to *multiTexCoord1d(Target, S)*.

`multiTexCoord1f(Target::enum(), S::float()) -> ok`

See **external** documentation.

`multiTexCoord1fv(Target, X2::{S}) -> ok`

Equivalent to *multiTexCoord1f(Target, S)*.

`multiTexCoord1i(Target::enum(), S::integer()) -> ok`

See **external** documentation.

`multiTexCoord1iv(Target, X2::{S}) -> ok`

Equivalent to *multiTexCoord1i(Target, S)*.

`multiTexCoord1s(Target::enum(), S::integer()) -> ok`

See **external** documentation.

`multiTexCoord1sv(Target, X2::{S}) -> ok`

Equivalent to *multiTexCoord1s(Target, S)*.

`multiTexCoord2d(Target::enum(), S::float(), T::float()) -> ok`

See **external** documentation.

`multiTexCoord2dv(Target, X2::{S, T}) -> ok`

Equivalent to *multiTexCoord2d(Target, S, T)*.

`multiTexCoord2f(Target::enum(), S::float(), T::float()) -> ok`

See **external** documentation.

`multiTexCoord2fv(Target, X2::{S, T}) -> ok`

Equivalent to *multiTexCoord2f(Target, S, T)*.

`multiTexCoord2i(Target::enum(), S::integer(), T::integer()) -> ok`

See **external** documentation.

---

```
multiTexCoord2iv(Target, X2::{S, T}) -> ok
```

Equivalent to *multiTexCoord2i(Target, S, T)*.

```
multiTexCoord2s(Target::enum(), S::integer(), T::integer()) -> ok
```

See **external** documentation.

```
multiTexCoord2sv(Target, X2::{S, T}) -> ok
```

Equivalent to *multiTexCoord2s(Target, S, T)*.

```
multiTexCoord3d(Target::enum(), S::float(), T::float(), R::float()) -> ok
```

See **external** documentation.

```
multiTexCoord3dv(Target, X2::{S, T, R}) -> ok
```

Equivalent to *multiTexCoord3d(Target, S, T, R)*.

```
multiTexCoord3f(Target::enum(), S::float(), T::float(), R::float()) -> ok
```

See **external** documentation.

```
multiTexCoord3fv(Target, X2::{S, T, R}) -> ok
```

Equivalent to *multiTexCoord3f(Target, S, T, R)*.

```
multiTexCoord3i(Target::enum(), S::integer(), T::integer(), R::integer()) -> ok
```

See **external** documentation.

```
multiTexCoord3iv(Target, X2::{S, T, R}) -> ok
```

Equivalent to *multiTexCoord3i(Target, S, T, R)*.

```
multiTexCoord3s(Target::enum(), S::integer(), T::integer(), R::integer()) -> ok
```

See **external** documentation.

```
multiTexCoord3sv(Target, X2::{S, T, R}) -> ok
```

Equivalent to *multiTexCoord3s(Target, S, T, R)*.

```
multiTexCoord4d(Target::enum(), S::float(), T::float(), R::float(),  
Q::float()) -> ok
```

See **external** documentation.

```
multiTexCoord4dv(Target, X2::{S, T, R, Q}) -> ok
```

Equivalent to *multiTexCoord4d(Target, S, T, R, Q)*.

```
multiTexCoord4f(Target::enum(), S::float(), T::float(), R::float(),  
Q::float()) -> ok
```

See **external** documentation.

```
multiTexCoord4fv(Target, X2::{S, T, R, Q}) -> ok
```

Equivalent to *multiTexCoord4f(Target, S, T, R, Q)*.

```
multiTexCoord4i(Target::enum(), S::integer(), T::integer(), R::integer(),  
Q::integer()) -> ok
```

See **external** documentation.

```
multiTexCoord4iv(Target, X2::{S, T, R, Q}) -> ok
```

Equivalent to *multiTexCoord4i(Target, S, T, R, Q)*.

```
multiTexCoord4s(Target::enum(), S::integer(), T::integer(), R::integer(),  
Q::integer()) -> ok
```

See **external** documentation.

```
multiTexCoord4sv(Target, X2::{S, T, R, Q}) -> ok
```

Equivalent to *multiTexCoord4s(Target, S, T, R, Q)*.

```
loadTransposeMatrixf(M::{float()}) -> ok
```

See **external** documentation.

```
loadTransposeMatrixd(M::{float()}) -> ok
```

See **external** documentation.

```
multTransposeMatrixf(M::{float()}) -> ok
```

See **external** documentation.

```
multTransposeMatrixd(M::{float()}) -> ok
```

See **external** documentation.

```
blendFuncSeparate(SfactorRGB::enum(), DfactorRGB::enum(),  
SfactorAlpha::enum(), DfactorAlpha::enum()) -> ok
```

See **external** documentation.

```
multiDrawArrays(Mode::enum(), First::[integer()], Count::[integer()]) -> ok
```

See **external** documentation.

```
pointParameterf(Pname::enum(), Param::float()) -> ok
```

See **external** documentation.



---

```
pointParameterfv(Pname::enum(), Params::{float()}) -> ok
```

See **external** documentation.

```
pointParameteri(Pname::enum(), Param::integer()) -> ok
```

See **external** documentation.

```
pointParameteriv(Pname::enum(), Params::{integer()}) -> ok
```

See **external** documentation.

```
fogCoordf(Coord::float()) -> ok
```

See **external** documentation.

```
fogCoordfv(X1::{Coord}) -> ok
```

Equivalent to *fogCoordf(Coord)*.

```
fogCoordd(Coord::float()) -> ok
```

See **external** documentation.

```
fogCoorddv(X1::{Coord}) -> ok
```

Equivalent to *fogCoordd(Coord)*.

```
fogCoordPointer(Type::enum(), Stride::integer(), Pointer::offset() |  
binary()) -> ok
```

See **external** documentation.

```
secondaryColor3b(Red::integer(), Green::integer(), Blue::integer()) -> ok
```

See **external** documentation.

```
secondaryColor3bv(X1::{Red, Green, Blue}) -> ok
```

Equivalent to *secondaryColor3b(Red, Green, Blue)*.

```
secondaryColor3d(Red::float(), Green::float(), Blue::float()) -> ok
```

See **external** documentation.

```
secondaryColor3dv(X1::{Red, Green, Blue}) -> ok
```

Equivalent to *secondaryColor3d(Red, Green, Blue)*.

```
secondaryColor3f(Red::float(), Green::float(), Blue::float()) -> ok
```

See **external** documentation.

```
secondaryColor3fv(X1::{Red, Green, Blue}) -> ok
```

Equivalent to *secondaryColor3f(Red, Green, Blue)*.

**secondaryColor3i**(Red::integer(), Green::integer(), Blue::integer()) -> ok

See **external** documentation.

**secondaryColor3iv**(X1::{Red, Green, Blue}) -> ok

Equivalent to *secondaryColor3i(Red, Green, Blue)*.

**secondaryColor3s**(Red::integer(), Green::integer(), Blue::integer()) -> ok

See **external** documentation.

**secondaryColor3sv**(X1::{Red, Green, Blue}) -> ok

Equivalent to *secondaryColor3s(Red, Green, Blue)*.

**secondaryColor3ub**(Red::integer(), Green::integer(), Blue::integer()) -> ok

See **external** documentation.

**secondaryColor3ubv**(X1::{Red, Green, Blue}) -> ok

Equivalent to *secondaryColor3ub(Red, Green, Blue)*.

**secondaryColor3ui**(Red::integer(), Green::integer(), Blue::integer()) -> ok

See **external** documentation.

**secondaryColor3uiv**(X1::{Red, Green, Blue}) -> ok

Equivalent to *secondaryColor3ui(Red, Green, Blue)*.

**secondaryColor3us**(Red::integer(), Green::integer(), Blue::integer()) -> ok

See **external** documentation.

**secondaryColor3usv**(X1::{Red, Green, Blue}) -> ok

Equivalent to *secondaryColor3us(Red, Green, Blue)*.

**secondaryColorPointer**(Size::integer(), Type::enum(), Stride::integer(),  
Pointer::offset() | binary()) -> ok

See **external** documentation.

**windowPos2d**(X::float(), Y::float()) -> ok

See **external** documentation.

**windowPos2dv**(X1::{X, Y}) -> ok

Equivalent to *windowPos2d(X, Y)*.

**windowPos2f**(X::float(), Y::float()) -> ok

See **external** documentation.

**windowPos2fv(X1::{X, Y}) -> ok**

Equivalent to *windowPos2f(X, Y)*.

**windowPos2i(X::integer(), Y::integer()) -> ok**

See **external** documentation.

**windowPos2iv(X1::{X, Y}) -> ok**

Equivalent to *windowPos2i(X, Y)*.

**windowPos2s(X::integer(), Y::integer()) -> ok**

See **external** documentation.

**windowPos2sv(X1::{X, Y}) -> ok**

Equivalent to *windowPos2s(X, Y)*.

**windowPos3d(X::float(), Y::float(), Z::float()) -> ok**

See **external** documentation.

**windowPos3dv(X1::{X, Y, Z}) -> ok**

Equivalent to *windowPos3d(X, Y, Z)*.

**windowPos3f(X::float(), Y::float(), Z::float()) -> ok**

See **external** documentation.

**windowPos3fv(X1::{X, Y, Z}) -> ok**

Equivalent to *windowPos3f(X, Y, Z)*.

**windowPos3i(X::integer(), Y::integer(), Z::integer()) -> ok**

See **external** documentation.

**windowPos3iv(X1::{X, Y, Z}) -> ok**

Equivalent to *windowPos3i(X, Y, Z)*.

**windowPos3s(X::integer(), Y::integer(), Z::integer()) -> ok**

See **external** documentation.

**windowPos3sv(X1::{X, Y, Z}) -> ok**

Equivalent to *windowPos3s(X, Y, Z)*.

**genQueries(N::integer()) -> [integer()]**

See **external** documentation.

**deleteQueries**(Ids:::[integer()]) -> ok

See **external** documentation.

**isQuery**(Id::integer()) -> 0 | 1

See **external** documentation.

**beginQuery**(Target::enum(), Id::integer()) -> ok

See **external** documentation.

**endQuery**(Target::enum()) -> ok

See **external** documentation.

**getQueryiv**(Target::enum(), Pname::enum()) -> integer()

See **external** documentation.

**getQueryObjectiv**(Id::integer(), Pname::enum()) -> integer()

See **external** documentation.

**getQueryObjectuiv**(Id::integer(), Pname::enum()) -> integer()

See **external** documentation.

**bindBuffer**(Target::enum(), Buffer::integer()) -> ok

See **external** documentation.

**deleteBuffers**(Buffers:::[integer()]) -> ok

See **external** documentation.

**genBuffers**(N::integer()) -> [integer()]

See **external** documentation.

**isBuffer**(Buffer::integer()) -> 0 | 1

See **external** documentation.

**bufferData**(Target::enum(), Size::integer(), Data::offset() | binary(),  
Usage::enum()) -> ok

See **external** documentation.

**bufferSubData**(Target::enum(), Offset::integer(), Size::integer(),  
Data::offset() | binary()) -> ok

See **external** documentation.

---

```
getBufferSubData(Target::enum(), Offset::integer(), Size::integer(),  
Data::wx_mem() (see module wx)) -> ok
```

See **external** documentation.

```
getBufferParameteriv(Target::enum(), Pname::enum()) -> integer()
```

See **external** documentation.

```
blendEquationSeparate(ModeRGB::enum(), ModeAlpha::enum()) -> ok
```

See **external** documentation.

```
drawBuffers(Bufs::[enum()]) -> ok
```

See **external** documentation.

```
stencilOpSeparate(Face::enum(), Sfail::enum(), Dpfail::enum(),  
Dppass::enum()) -> ok
```

See **external** documentation.

```
stencilFuncSeparate(Frontfunc::enum(), Backfunc::enum(), Ref::integer(),  
Mask::integer()) -> ok
```

See **external** documentation.

```
stencilMaskSeparate(Face::enum(), Mask::integer()) -> ok
```

See **external** documentation.

```
attachShader(Program::integer(), Shader::integer()) -> ok
```

See **external** documentation.

```
bindAttribLocation(Program::integer(), Index::integer(), Name::string()) ->  
ok
```

See **external** documentation.

```
compileShader(Shader::integer()) -> ok
```

See **external** documentation.

```
createProgram() -> integer()
```

See **external** documentation.

```
createShader(Type::enum()) -> integer()
```

See **external** documentation.

```
deleteProgram(Program::integer()) -> ok
```

See **external** documentation.

`deleteShader(Shader::integer()) -> ok`

See **external** documentation.

`detachShader(Program::integer(), Shader::integer()) -> ok`

See **external** documentation.

`disableVertexAttribArray(Index::integer()) -> ok`

See **external** documentation.

`enableVertexAttribArray(Index::integer()) -> ok`

See **external** documentation.

`getActiveAttrib(Program::integer(), Index::integer(), BufSize::integer()) -> {Size::integer(), Type::enum(), Name::string()}`

See **external** documentation.

`getActiveUniform(Program::integer(), Index::integer(), BufSize::integer()) -> {Size::integer(), Type::enum(), Name::string()}`

See **external** documentation.

`getAttachedShaders(Program::integer(), MaxCount::integer()) -> [integer()]`

See **external** documentation.

`getAttribLocation(Program::integer(), Name::string()) -> integer()`

See **external** documentation.

`getProgramiv(Program::integer(), Pname::enum()) -> integer()`

See **external** documentation.

`getProgramInfoLog(Program::integer(), BufSize::integer()) -> string()`

See **external** documentation.

`getShaderiv(Shader::integer(), Pname::enum()) -> integer()`

See **external** documentation.

`getShaderInfoLog(Shader::integer(), BufSize::integer()) -> string()`

See **external** documentation.

`getShaderSource(Shader::integer(), BufSize::integer()) -> string()`

See **external** documentation.

`getUniformLocation(Program::integer(), Name::string()) -> integer()`

See **external** documentation.

---

```
getUniformfv(Program::integer(), Location::integer()) -> {float()}
```

See **external** documentation.

```
getUniformiv(Program::integer(), Location::integer()) -> {integer()}
```

See **external** documentation.

```
getVertexAttribdv(Index::integer(), Pname::enum()) -> {float()}
```

See **external** documentation.

```
getVertexAttribfv(Index::integer(), Pname::enum()) -> {float()}
```

See **external** documentation.

```
getVertexAttribiv(Index::integer(), Pname::enum()) -> {integer()}
```

See **external** documentation.

```
isProgram(Program::integer()) -> 0 | 1
```

See **external** documentation.

```
isShader(Shader::integer()) -> 0 | 1
```

See **external** documentation.

```
linkProgram(Program::integer()) -> ok
```

See **external** documentation.

```
shaderSource(Shader::integer(), String::[string()]) -> ok
```

See **external** documentation.

```
useProgram(Program::integer()) -> ok
```

See **external** documentation.

```
uniform1f(Location::integer(), V0::float()) -> ok
```

See **external** documentation.

```
uniform2f(Location::integer(), V0::float(), V1::float()) -> ok
```

See **external** documentation.

```
uniform3f(Location::integer(), V0::float(), V1::float(), V2::float()) -> ok
```

See **external** documentation.

```
uniform4f(Location::integer(), V0::float(), V1::float(), V2::float(),  
V3::float()) -> ok
```

See **external** documentation.

`uniform1i(Location::integer(), V0::integer()) -> ok`

See **external** documentation.

`uniform2i(Location::integer(), V0::integer(), V1::integer()) -> ok`

See **external** documentation.

`uniform3i(Location::integer(), V0::integer(), V1::integer(), V2::integer()) -> ok`

See **external** documentation.

`uniform4i(Location::integer(), V0::integer(), V1::integer(), V2::integer(), V3::integer()) -> ok`

See **external** documentation.

`uniform1fv(Location::integer(), Value::[float()]) -> ok`

See **external** documentation.

`uniform2fv(Location::integer(), Value::[{float()}]) -> ok`

See **external** documentation.

`uniform3fv(Location::integer(), Value::[{float()}]) -> ok`

See **external** documentation.

`uniform4fv(Location::integer(), Value::[{float()}]) -> ok`

See **external** documentation.

`uniform1iv(Location::integer(), Value::[integer()]) -> ok`

See **external** documentation.

`uniform2iv(Location::integer(), Value::[{integer()}]) -> ok`

See **external** documentation.

`uniform3iv(Location::integer(), Value::[{integer()}]) -> ok`

See **external** documentation.

`uniform4iv(Location::integer(), Value::[{integer()}]) -> ok`

See **external** documentation.

`uniformMatrix2fv(Location::integer(), Transpose::0 | 1, Value::[{float()}]) -> ok`

See **external** documentation.



---

```
uniformMatrix3fv(Location::integer(), Transpose::0 | 1, Value::[{float()}]) -  
> ok
```

See **external** documentation.

```
uniformMatrix4fv(Location::integer(), Transpose::0 | 1, Value::[{float()}]) -  
> ok
```

See **external** documentation.

```
validateProgram(Program::integer()) -> ok
```

See **external** documentation.

```
vertexAttrib1d(Index::integer(), X::float()) -> ok
```

See **external** documentation.

```
vertexAttrib1dv(Index, X2::{X}) -> ok
```

Equivalent to *vertexAttrib1d(Index, X)*.

```
vertexAttrib1f(Index::integer(), X::float()) -> ok
```

See **external** documentation.

```
vertexAttrib1fv(Index, X2::{X}) -> ok
```

Equivalent to *vertexAttrib1f(Index, X)*.

```
vertexAttrib1s(Index::integer(), X::integer()) -> ok
```

See **external** documentation.

```
vertexAttrib1sv(Index, X2::{X}) -> ok
```

Equivalent to *vertexAttrib1s(Index, X)*.

```
vertexAttrib2d(Index::integer(), X::float(), Y::float()) -> ok
```

See **external** documentation.

```
vertexAttrib2dv(Index, X2::{X, Y}) -> ok
```

Equivalent to *vertexAttrib2d(Index, X, Y)*.

```
vertexAttrib2f(Index::integer(), X::float(), Y::float()) -> ok
```

See **external** documentation.

```
vertexAttrib2fv(Index, X2::{X, Y}) -> ok
```

Equivalent to *vertexAttrib2f(Index, X, Y)*.

```
vertexAttrib2s(Index::integer(), X::integer(), Y::integer()) -> ok
```

See **external** documentation.

**vertexAttrib2sv**(Index, X2::{X, Y}) -> ok

Equivalent to *vertexAttrib2s*(Index, X, Y).

**vertexAttrib3d**(Index::integer(), X::float(), Y::float(), Z::float()) -> ok

See **external** documentation.

**vertexAttrib3dv**(Index, X2::{X, Y, Z}) -> ok

Equivalent to *vertexAttrib3d*(Index, X, Y, Z).

**vertexAttrib3f**(Index::integer(), X::float(), Y::float(), Z::float()) -> ok

See **external** documentation.

**vertexAttrib3fv**(Index, X2::{X, Y, Z}) -> ok

Equivalent to *vertexAttrib3f*(Index, X, Y, Z).

**vertexAttrib3s**(Index::integer(), X::integer(), Y::integer(), Z::integer()) -> ok

See **external** documentation.

**vertexAttrib3sv**(Index, X2::{X, Y, Z}) -> ok

Equivalent to *vertexAttrib3s*(Index, X, Y, Z).

**vertexAttrib4Nbv**(Index::integer(), V::{integer()}) -> ok

See **external** documentation.

**vertexAttrib4Niv**(Index::integer(), V::{integer()}) -> ok

See **external** documentation.

**vertexAttrib4Nsv**(Index::integer(), V::{integer()}) -> ok

See **external** documentation.

**vertexAttrib4Nub**(Index::integer(), X::integer(), Y::integer(), Z::integer(), W::integer()) -> ok

See **external** documentation.

**vertexAttrib4Nubv**(Index, X2::{X, Y, Z, W}) -> ok

Equivalent to *vertexAttrib4Nub*(Index, X, Y, Z, W).

**vertexAttrib4Nuiv**(Index::integer(), V::{integer()}) -> ok

See **external** documentation.

**vertexAttrib4Nusv**(Index::integer(), V::{integer()}) -> ok

See **external** documentation.

---

```
vertexAttrib4bv(Index::integer(), V::{integer()}) -> ok
```

See **external** documentation.

```
vertexAttrib4d(Index::integer(), X::float(), Y::float(), Z::float(),  
W::float()) -> ok
```

See **external** documentation.

```
vertexAttrib4dv(Index, X2::{X, Y, Z, W}) -> ok
```

Equivalent to *vertexAttrib4d(Index, X, Y, Z, W)*.

```
vertexAttrib4f(Index::integer(), X::float(), Y::float(), Z::float(),  
W::float()) -> ok
```

See **external** documentation.

```
vertexAttrib4fv(Index, X2::{X, Y, Z, W}) -> ok
```

Equivalent to *vertexAttrib4f(Index, X, Y, Z, W)*.

```
vertexAttrib4iv(Index::integer(), V::{integer()}) -> ok
```

See **external** documentation.

```
vertexAttrib4s(Index::integer(), X::integer(), Y::integer(), Z::integer(),  
W::integer()) -> ok
```

See **external** documentation.

```
vertexAttrib4sv(Index, X2::{X, Y, Z, W}) -> ok
```

Equivalent to *vertexAttrib4s(Index, X, Y, Z, W)*.

```
vertexAttrib4ubv(Index::integer(), V::{integer()}) -> ok
```

See **external** documentation.

```
vertexAttrib4uiv(Index::integer(), V::{integer()}) -> ok
```

See **external** documentation.

```
vertexAttrib4usv(Index::integer(), V::{integer()}) -> ok
```

See **external** documentation.

```
vertexAttribPointer(Index::integer(), Size::integer(), Type::enum(),  
Normalized::0 | 1, Stride::integer(), Pointer::offset() | binary()) -> ok
```

See **external** documentation.

```
uniformMatrix2x3fv(Location::integer(), Transpose::0 | 1, Value::{float()})  
-> ok
```

See **external** documentation.

```
uniformMatrix3x2fv(Location::integer(), Transpose::0 | 1, Value::[{float()}])  
-> ok
```

See **external** documentation.

```
uniformMatrix2x4fv(Location::integer(), Transpose::0 | 1, Value::[{float()}])  
-> ok
```

See **external** documentation.

```
uniformMatrix4x2fv(Location::integer(), Transpose::0 | 1, Value::[{float()}])  
-> ok
```

See **external** documentation.

```
uniformMatrix3x4fv(Location::integer(), Transpose::0 | 1, Value::[{float()}])  
-> ok
```

See **external** documentation.

```
uniformMatrix4x3fv(Location::integer(), Transpose::0 | 1, Value::[{float()}])  
-> ok
```

See **external** documentation.

```
colorMaski(Index::integer(), R::0 | 1, G::0 | 1, B::0 | 1, A::0 | 1) -> ok
```

See **external** documentation.

```
getBooleani_v(Target::enum(), Index::integer()) -> [0 | 1]
```

See **external** documentation.

```
getIntegeri_v(Target::enum(), Index::integer()) -> [integer()]
```

See **external** documentation.

```
enablei(Target::enum(), Index::integer()) -> ok
```

See **external** documentation.

```
disablei(Target::enum(), Index::integer()) -> ok
```

See **external** documentation.

```
isEnabledi(Target::enum(), Index::integer()) -> 0 | 1
```

See **external** documentation.

```
beginTransformFeedback(PrimitiveMode::enum()) -> ok
```

See **external** documentation.

```
endTransformFeedback() -> ok
```

See **external** documentation.

---

```
bindBufferRange(Target::enum(), Index::integer(), Buffer::integer(),  
Offset::integer(), Size::integer()) -> ok
```

See **external** documentation.

```
bindBufferBase(Target::enum(), Index::integer(), Buffer::integer()) -> ok
```

See **external** documentation.

```
transformFeedbackVaryings(Program::integer(), Varyings::[string()],  
BufferMode::enum()) -> ok
```

See **external** documentation.

```
getTransformFeedbackVarying(Program::integer(), Index::integer(),  
BufSize::integer()) -> {Size::integer(), Type::enum(), Name::string()}
```

See **external** documentation.

```
clampColor(Target::enum(), Clamp::enum()) -> ok
```

See **external** documentation.

```
beginConditionalRender(Id::integer(), Mode::enum()) -> ok
```

See **external** documentation.

```
endConditionalRender() -> ok
```

See **external** documentation.

```
vertexAttribIPointer(Index::integer(), Size::integer(), Type::enum(),  
Stride::integer(), Pointer::offset() | binary()) -> ok
```

See **external** documentation.

```
getVertexAttribIiv(Index::integer(), Pname::enum()) -> {integer()}
```

See **external** documentation.

```
getVertexAttribIuiv(Index::integer(), Pname::enum()) -> {integer()}
```

See **external** documentation.

```
getUniformuiv(Program::integer(), Location::integer()) -> {integer()}
```

See **external** documentation.

```
bindFragDataLocation(Program::integer(), Color::integer(), Name::string()) ->  
ok
```

See **external** documentation.

```
getFragDataLocation(Program::integer(), Name::string()) -> integer()
```

See **external** documentation.

`uniform1ui(Location::integer(), V0::integer()) -> ok`

See **external** documentation.

`uniform2ui(Location::integer(), V0::integer(), V1::integer()) -> ok`

See **external** documentation.

`uniform3ui(Location::integer(), V0::integer(), V1::integer(), V2::integer())  
-> ok`

See **external** documentation.

`uniform4ui(Location::integer(), V0::integer(), V1::integer(), V2::integer(),  
V3::integer()) -> ok`

See **external** documentation.

`uniform1uiv(Location::integer(), Value::[integer()]) -> ok`

See **external** documentation.

`uniform2uiv(Location::integer(), Value::[{integer()}]) -> ok`

See **external** documentation.

`uniform3uiv(Location::integer(), Value::[{integer()}]) -> ok`

See **external** documentation.

`uniform4uiv(Location::integer(), Value::[{integer()}]) -> ok`

See **external** documentation.

`texParameterIiv(Target::enum(), Pname::enum(), Params::{integer()}) -> ok`

See **external** documentation.

`texParameterIuiv(Target::enum(), Pname::enum(), Params::{integer()}) -> ok`

See **external** documentation.

`getTexParameterIiv(Target::enum(), Pname::enum()) -> {integer()}`

See **external** documentation.

`getTexParameterIuiv(Target::enum(), Pname::enum()) -> {integer()}`

See **external** documentation.

`clearBufferiv(Buffer::enum(), Drawbuffer::integer(), Value::{integer()}) ->  
ok`

See **external** documentation.

---

```
clearBufferuiv(Buffer::enum(), Drawbuffer::integer(), Value::{integer()}) ->
ok
```

See **external** documentation.

```
clearBufferfv(Buffer::enum(), Drawbuffer::integer(), Value::{float()}) -> ok
```

See **external** documentation.

```
clearBufferfi(Buffer::enum(), Drawbuffer::integer(), Depth::float(),
Stencil::integer()) -> ok
```

See **external** documentation.

```
getStringi(Name::enum(), Index::integer()) -> string()
```

See **external** documentation.

```
vertexAttribI1i(Index::integer(), X::integer()) -> ok
```

See **external** documentation.

```
vertexAttribI2i(Index::integer(), X::integer(), Y::integer()) -> ok
```

See **external** documentation.

```
vertexAttribI3i(Index::integer(), X::integer(), Y::integer(), Z::integer()) -
> ok
```

See **external** documentation.

```
vertexAttribI4i(Index::integer(), X::integer(), Y::integer(), Z::integer(),
W::integer()) -> ok
```

See **external** documentation.

```
vertexAttribI1ui(Index::integer(), X::integer()) -> ok
```

See **external** documentation.

```
vertexAttribI2ui(Index::integer(), X::integer(), Y::integer()) -> ok
```

See **external** documentation.

```
vertexAttribI3ui(Index::integer(), X::integer(), Y::integer(), Z::integer())
-> ok
```

See **external** documentation.

```
vertexAttribI4ui(Index::integer(), X::integer(), Y::integer(), Z::integer(),
W::integer()) -> ok
```

See **external** documentation.

```
vertexAttribI1iv(Index, X2::{X}) -> ok
```

Equivalent to *vertexAttribI1i(Index, X)*.

**vertexAttribI2iv(Index, X2::{X, Y}) -> ok**

Equivalent to *vertexAttribI2i(Index, X, Y)*.

**vertexAttribI3iv(Index, X2::{X, Y, Z}) -> ok**

Equivalent to *vertexAttribI3i(Index, X, Y, Z)*.

**vertexAttribI4iv(Index, X2::{X, Y, Z, W}) -> ok**

Equivalent to *vertexAttribI4i(Index, X, Y, Z, W)*.

**vertexAttribI1uiv(Index, X2::{X}) -> ok**

Equivalent to *vertexAttribI1ui(Index, X)*.

**vertexAttribI2uiv(Index, X2::{X, Y}) -> ok**

Equivalent to *vertexAttribI2ui(Index, X, Y)*.

**vertexAttribI3uiv(Index, X2::{X, Y, Z}) -> ok**

Equivalent to *vertexAttribI3ui(Index, X, Y, Z)*.

**vertexAttribI4uiv(Index, X2::{X, Y, Z, W}) -> ok**

Equivalent to *vertexAttribI4ui(Index, X, Y, Z, W)*.

**vertexAttribI4bv(Index::integer(), V::{integer()}) -> ok**

See **external** documentation.

**vertexAttribI4sv(Index::integer(), V::{integer()}) -> ok**

See **external** documentation.

**vertexAttribI4ubv(Index::integer(), V::{integer()}) -> ok**

See **external** documentation.

**vertexAttribI4usv(Index::integer(), V::{integer()}) -> ok**

See **external** documentation.

**drawArraysInstanced(Mode::enum(), First::integer(), Count::integer(),  
Primcount::integer()) -> ok**

See **external** documentation.

**drawElementsInstanced(Mode::enum(), Count::integer(), Type::enum(),  
Indices::offset() | binary(), Primcount::integer()) -> ok**

See **external** documentation.

**texBuffer(Target::enum(), Internalformat::enum(), Buffer::integer()) -> ok**

See **external** documentation.



---

**primitiveRestartIndex**(Index::integer()) -> ok

See **external** documentation.

**loadTransposeMatrixfARB**(M::{float()}) -> ok

See **external** documentation.

**loadTransposeMatrixdARB**(M::{float()}) -> ok

See **external** documentation.

**multTransposeMatrixfARB**(M::{float()}) -> ok

See **external** documentation.

**multTransposeMatrixdARB**(M::{float()}) -> ok

See **external** documentation.

**weightbvARB**(Weights::[integer()]) -> ok

See **external** documentation.

**weightsvARB**(Weights::[integer()]) -> ok

See **external** documentation.

**weightivARB**(Weights::[integer()]) -> ok

See **external** documentation.

**weightfvARB**(Weights::[float()]) -> ok

See **external** documentation.

**weightdvARB**(Weights::[float()]) -> ok

See **external** documentation.

**weightubvARB**(Weights::[integer()]) -> ok

See **external** documentation.

**weightusvARB**(Weights::[integer()]) -> ok

See **external** documentation.

**weightuivARB**(Weights::[integer()]) -> ok

See **external** documentation.

**vertexBlendARB**(Count::integer()) -> ok

See **external** documentation.

**currentPaletteMatrixARB(Index::integer()) -> ok**

See **external** documentation.

**matrixIndexubvARB(Indices::[integer()]) -> ok**

See **external** documentation.

**matrixIndexusvARB(Indices::[integer()]) -> ok**

See **external** documentation.

**matrixIndexuivARB(Indices::[integer()]) -> ok**

See **external** documentation.

**programStringARB(Target::enum(), Format::enum(), String::string()) -> ok**

See **external** documentation.

**bindProgramARB(Target::enum(), Program::integer()) -> ok**

See **external** documentation.

**deleteProgramsARB(Programs::[integer()]) -> ok**

See **external** documentation.

**genProgramsARB(N::integer()) -> [integer()]**

See **external** documentation.

**programEnvParameter4dARB(Target::enum(), Index::integer(), X::float(), Y::float(), Z::float(), W::float()) -> ok**

See **external** documentation.

**programEnvParameter4dvARB(Target::enum(), Index::integer(), Params::  
{float()}) -> ok**

See **external** documentation.

**programEnvParameter4fARB(Target::enum(), Index::integer(), X::float(), Y::float(), Z::float(), W::float()) -> ok**

See **external** documentation.

**programEnvParameter4fvARB(Target::enum(), Index::integer(), Params::  
{float()}) -> ok**

See **external** documentation.

**programLocalParameter4dARB(Target::enum(), Index::integer(), X::float(), Y::float(), Z::float(), W::float()) -> ok**

See **external** documentation.

---

```
programLocalParameter4dvARB(Target::enum(), Index::integer(), Params::
{float()}) -> ok
```

See **external** documentation.

```
programLocalParameter4fARB(Target::enum(), Index::integer(), X::float(),
Y::float(), Z::float(), W::float()) -> ok
```

See **external** documentation.

```
programLocalParameter4fvARB(Target::enum(), Index::integer(), Params::
{float()}) -> ok
```

See **external** documentation.

```
getProgramEnvParameterdvARB(Target::enum(), Index::integer()) -> {float()}
```

See **external** documentation.

```
getProgramEnvParameterfvARB(Target::enum(), Index::integer()) -> {float()}
```

See **external** documentation.

```
getProgramLocalParameterdvARB(Target::enum(), Index::integer()) -> {float()}
```

See **external** documentation.

```
getProgramLocalParameterfvARB(Target::enum(), Index::integer()) -> {float()}
```

See **external** documentation.

```
getProgramStringARB(Target::enum(), Pname::enum(), String::wx_mem() (see
module wx)) -> ok
```

See **external** documentation.

```
deleteObjectARB(Obj::integer()) -> ok
```

See **external** documentation.

```
getHandleARB(Pname::enum()) -> integer()
```

See **external** documentation.

```
detachObjectARB(ContainerObj::integer(), AttachedObj::integer()) -> ok
```

See **external** documentation.

```
createShaderObjectARB(ShaderType::enum()) -> integer()
```

See **external** documentation.

```
shaderSourceARB(ShaderObj::integer(), String::[string()]) -> ok
```

See **external** documentation.

`compileShaderARB(ShaderObj::integer()) -> ok`

See **external** documentation.

`createProgramObjectARB() -> integer()`

See **external** documentation.

`attachObjectARB(ContainerObj::integer(), Obj::integer()) -> ok`

See **external** documentation.

`linkProgramARB(ProgramObj::integer()) -> ok`

See **external** documentation.

`useProgramObjectARB(ProgramObj::integer()) -> ok`

See **external** documentation.

`validateProgramARB(ProgramObj::integer()) -> ok`

See **external** documentation.

`getObjectParameterfvARB(Obj::integer(), Pname::enum()) -> float()`

See **external** documentation.

`getObjectParameterivARB(Obj::integer(), Pname::enum()) -> integer()`

See **external** documentation.

`getInfoLogARB(Obj::integer(), MaxLength::integer()) -> string()`

See **external** documentation.

`getAttachedObjectsARB(ContainerObj::integer(), MaxCount::integer()) -> [integer()]`

See **external** documentation.

`getUniformLocationARB(ProgramObj::integer(), Name::string()) -> integer()`

See **external** documentation.

`getActiveUniformARB(ProgramObj::integer(), Index::integer(),  
MaxLength::integer()) -> {Size::integer(), Type::enum(), Name::string()}`

See **external** documentation.

`getUniformfvARB(ProgramObj::integer(), Location::integer()) -> {float()}`

See **external** documentation.

`getUniformivARB(ProgramObj::integer(), Location::integer()) -> {integer()}`

See **external** documentation.

---

```
getShaderSourceARB(Obj::integer(), MaxLength::integer()) -> string()
```

See **external** documentation.

```
bindAttribLocationARB(ProgramObj::integer(), Index::integer(),  
Name::string()) -> ok
```

See **external** documentation.

```
getActiveAttribARB(ProgramObj::integer(), Index::integer(),  
MaxLength::integer()) -> {Size::integer(), Type::enum(), Name::string()}
```

See **external** documentation.

```
getAttribLocationARB(ProgramObj::integer(), Name::string()) -> integer()
```

See **external** documentation.

```
isRenderbuffer(Renderbuffer::integer()) -> 0 | 1
```

See **external** documentation.

```
bindRenderbuffer(Target::enum(), Renderbuffer::integer()) -> ok
```

See **external** documentation.

```
deleteRenderbuffers(Renderbuffers::[integer()]) -> ok
```

See **external** documentation.

```
genRenderbuffers(N::integer()) -> [integer()]
```

See **external** documentation.

```
renderbufferStorage(Target::enum(), Internalformat::enum(), Width::integer(),  
Height::integer()) -> ok
```

See **external** documentation.

```
getRenderbufferParameteriv(Target::enum(), Pname::enum()) -> integer()
```

See **external** documentation.

```
isFramebuffer(Framebuffer::integer()) -> 0 | 1
```

See **external** documentation.

```
bindFramebuffer(Target::enum(), Framebuffer::integer()) -> ok
```

See **external** documentation.

```
deleteFramebuffers(Framebuffers::[integer()]) -> ok
```

See **external** documentation.

**genFramebuffers**(N::integer()) -> [integer()]

See **external** documentation.

**checkFramebufferStatus**(Target::enum()) -> enum()

See **external** documentation.

**framebufferTexture1D**(Target::enum(), Attachment::enum(), Textarget::enum(), Texture::integer(), Level::integer()) -> ok

See **external** documentation.

**framebufferTexture2D**(Target::enum(), Attachment::enum(), Textarget::enum(), Texture::integer(), Level::integer()) -> ok

See **external** documentation.

**framebufferTexture3D**(Target::enum(), Attachment::enum(), Textarget::enum(), Texture::integer(), Level::integer(), Zoffset::integer()) -> ok

See **external** documentation.

**framebufferRenderbuffer**(Target::enum(), Attachment::enum(), Renderbuffertarget::enum(), Renderbuffer::integer()) -> ok

See **external** documentation.

**getFramebufferAttachmentParameteriv**(Target::enum(), Attachment::enum(), Pname::enum()) -> integer()

See **external** documentation.

**generateMipmap**(Target::enum()) -> ok

See **external** documentation.

**blitFramebuffer**(SrcX0::integer(), SrcY0::integer(), SrcX1::integer(), SrcY1::integer(), DstX0::integer(), DstY0::integer(), DstX1::integer(), DstY1::integer(), Mask::integer(), Filter::enum()) -> ok

See **external** documentation.

**renderbufferStorageMultisample**(Target::enum(), Samples::integer(), Internalformat::enum(), Width::integer(), Height::integer()) -> ok

See **external** documentation.

**framebufferTextureLayer**(Target::enum(), Attachment::enum(), Texture::integer(), Level::integer(), Layer::integer()) -> ok

See **external** documentation.

---

```
programParameteriARB(Program::integer(), Pname::enum(), Value::integer()) ->
ok
```

See **external** documentation.

```
framebufferTextureARB(Target::enum(), Attachment::enum(), Texture::integer(),
Level::integer()) -> ok
```

See **external** documentation.

```
framebufferTextureFaceARB(Target::enum(), Attachment::enum(),
Texture::integer(), Level::integer(), Face::enum()) -> ok
```

See **external** documentation.

```
vertexAttribDivisorARB(Index::integer(), Divisor::integer()) -> ok
```

See **external** documentation.

```
flushMappedBufferRange(Target::enum(), Offset::integer(), Length::integer())
-> ok
```

See **external** documentation.

```
bindVertexArray(Array::integer()) -> ok
```

See **external** documentation.

```
deleteVertexArrays(Arrays::[integer()]) -> ok
```

See **external** documentation.

```
genVertexArrays(N::integer()) -> [integer()]
```

See **external** documentation.

```
isVertexArray(Array::integer()) -> 0 | 1
```

See **external** documentation.

```
getUniformIndices(Program::integer(), UniformNames::[string()]) ->
[integer()]
```

See **external** documentation.

```
getActiveUniformsiv(Program::integer(), UniformIndices::[integer()],
Pname::enum()) -> [integer()]
```

See **external** documentation.

```
getActiveUniformName(Program::integer(), UniformIndex::integer(),
BufSize::integer()) -> string()
```

See **external** documentation.

```
getUniformBlockIndex(Program::integer(), UniformBlockName::string()) ->
integer()
```

See **external** documentation.

```
getActiveUniformBlockiv(Program::integer(), UniformBlockIndex::integer(),
Pname::enum(), Params::wx_mem() (see module wx)) -> ok
```

See **external** documentation.

```
getActiveUniformBlockName(Program::integer(), UniformBlockIndex::integer(),
BufSize::integer()) -> string()
```

See **external** documentation.

```
uniformBlockBinding(Program::integer(), UniformBlockIndex::integer(),
UniformBlockBinding::integer()) -> ok
```

See **external** documentation.

```
copyBufferSubData(ReadTarget::enum(), WriteTarget::enum(),
ReadOffset::integer(), WriteOffset::integer(), Size::integer()) -> ok
```

See **external** documentation.

```
resizeBuffersMESA() -> ok
```

See **external** documentation.

```
windowPos4dMESA(X::float(), Y::float(), Z::float(), W::float()) -> ok
```

See **external** documentation.

```
windowPos4dvMESA(X1::{X, Y, Z, W}) -> ok
```

Equivalent to *windowPos4dMESA(X, Y, Z, W)*.

```
windowPos4fMESA(X::float(), Y::float(), Z::float(), W::float()) -> ok
```

See **external** documentation.

```
windowPos4fvMESA(X1::{X, Y, Z, W}) -> ok
```

Equivalent to *windowPos4fMESA(X, Y, Z, W)*.

```
windowPos4iMESA(X::integer(), Y::integer(), Z::integer(), W::integer()) -> ok
```

See **external** documentation.

```
windowPos4ivMESA(X1::{X, Y, Z, W}) -> ok
```

Equivalent to *windowPos4iMESA(X, Y, Z, W)*.

```
windowPos4sMESA(X::integer(), Y::integer(), Z::integer(), W::integer()) -> ok
```

See **external** documentation.



**windowPos4svMESA(X1::{X, Y, Z, W}) -> ok**

Equivalent to *windowPos4sMESA(X, Y, Z, W)*.

**depthBoundsEXT(Zmin::clamp(), Zmax::clamp()) -> ok**

See **external** documentation.

**stencilClearTagEXT(StencilTagBits::integer(), StencilClearTag::integer()) -> ok**

See **external** documentation.