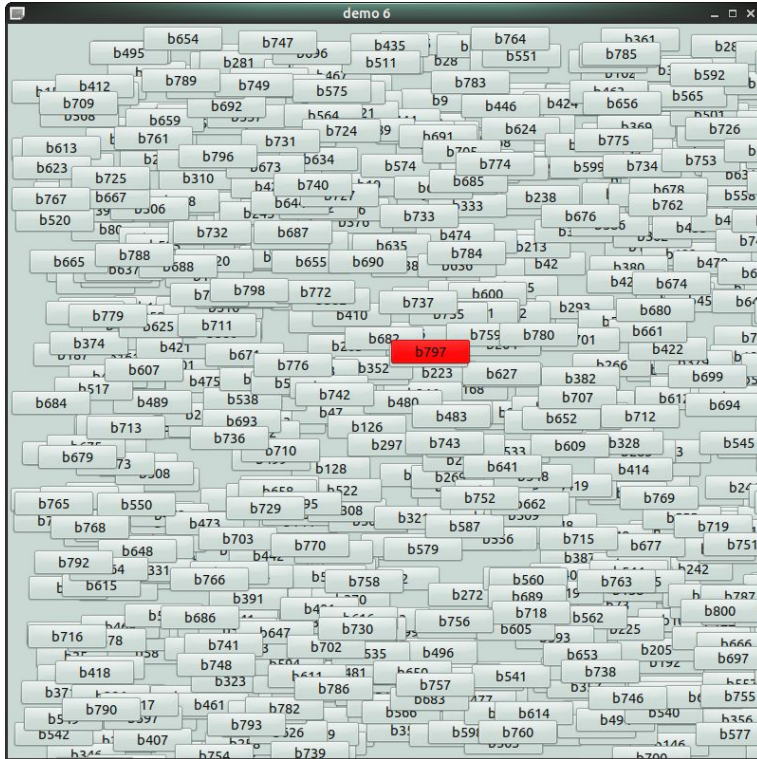




# concurrency and wxErlang

an introduction, by Arif Ishaq

# what to expect from this talk



- spawn a process per widget
- supervise widget processes
- let widget processes crash
- demystify wx\_object



# how I got here

## Guides

- <https://arifishaq.files.wordpress.com/2017/12/wxerlang-getting-started.pdf>
- <https://arifishaq.files.wordpress.com/2018/04/wxerlang-speeding-up.pdf>



# wxWidgets

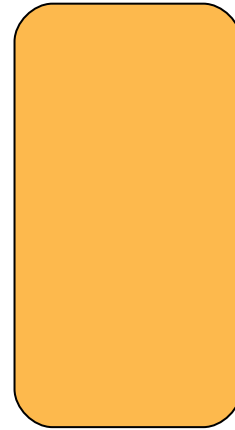
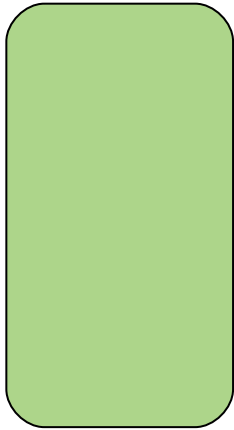
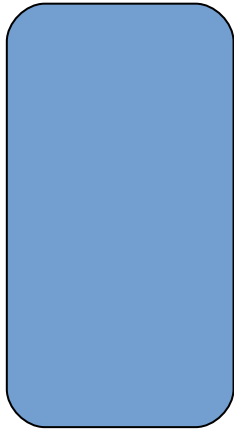
- Cross-platform **C++** library of widgets
- **Native look and feel**



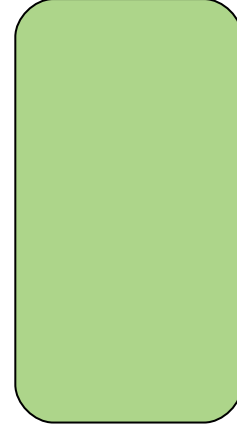
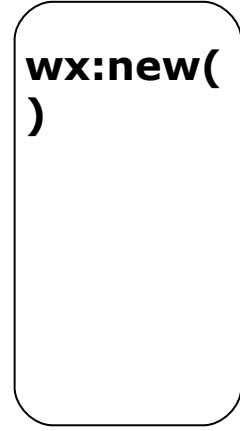
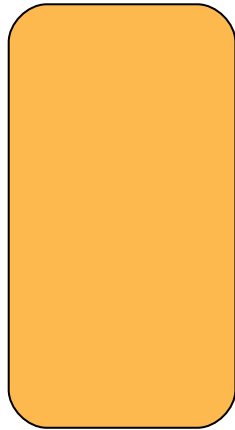
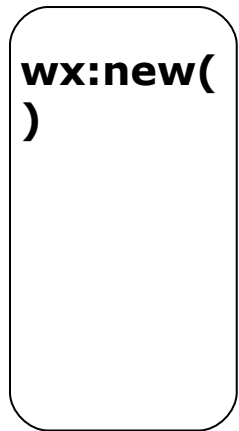
# wxErlang

- **wx** module = wxWidgets binding
- huge, with auto-generated documentation
- refer to **wxWidgets** documentation for the **meaning**
- refer to **wx** documentation for the **method signatures**

**processes either have a wx environment or they don't**

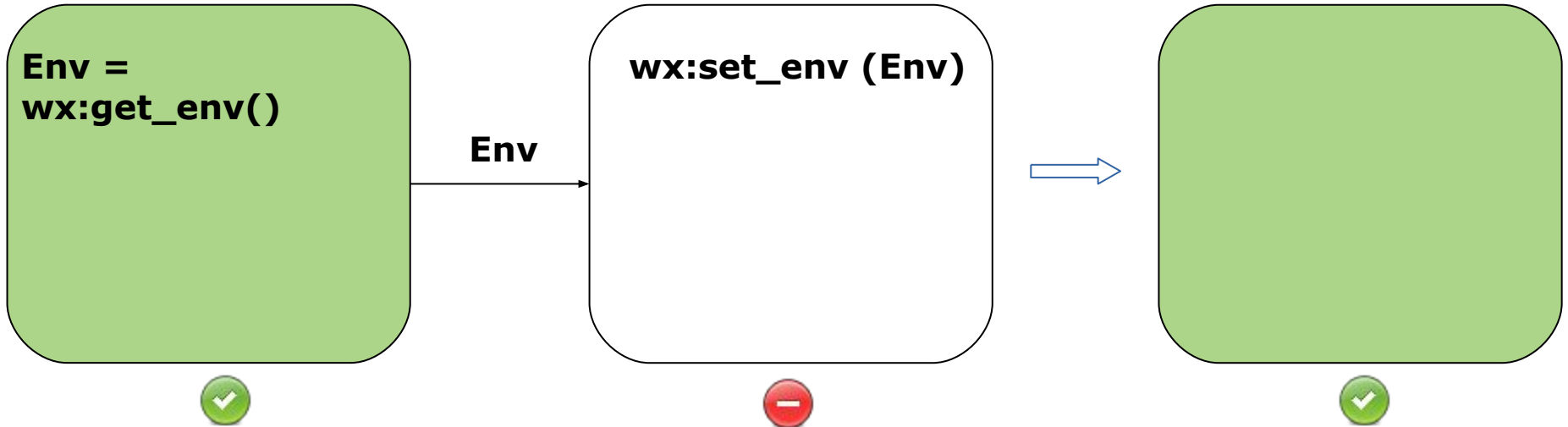


# either create the environment ...



**each wx:new/0  
creates a different  
environment**

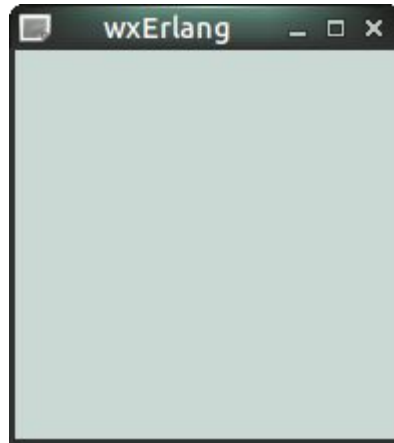
or get it from some other process





# why do we need the environment?

```
wxFrame:new  
(  
  wx:null(),  
  ?wxID_ANY,  
  "wxErlang"  
)
```



```
wxFrame:new(  
  .)
```



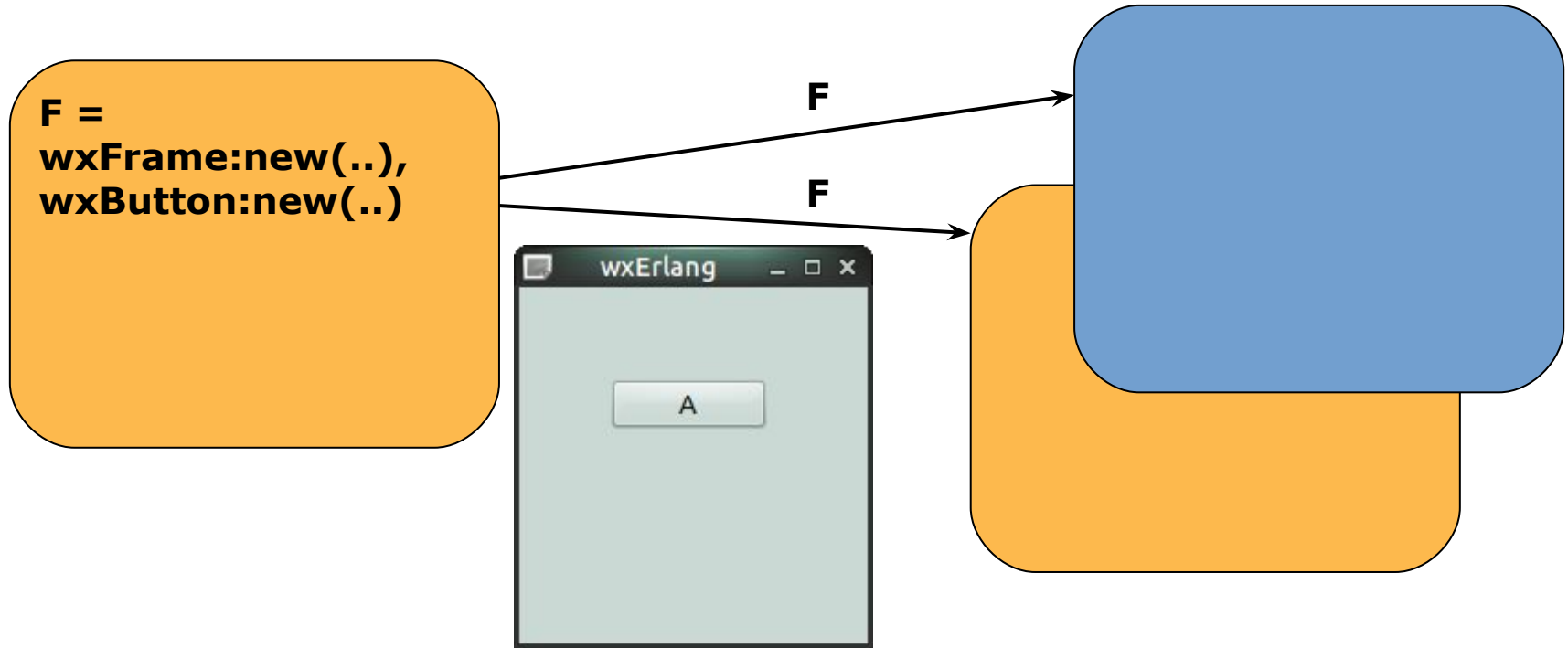
# children need a parent

```
F =  
wxFrame:new(..),  
  
wxButton:new(  
  F, ?wxID_ANY,  
  [{label, "A"}])
```



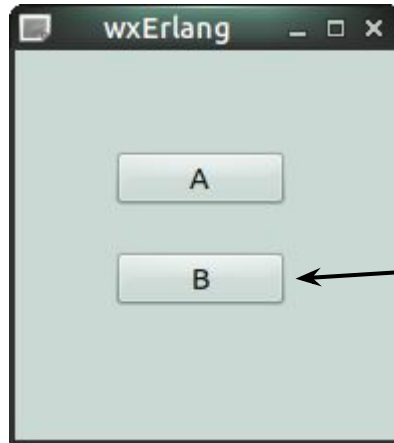
the frame F is the  
parent

# references can be passed to others



# but work only in the same environment

```
F =  
wxFrame:new(..),  
wxButton:new(..)
```



```
wxButton:new(  
F, ?wxID_ANY,  
[{label, "C"}])
```



```
wxButton:new(  
F, ?wxID_ANY,  
[{label, "B"}])
```

# parent can create child widget by spawning a process

```
F = wxFrame:new(..),  
{ok,ChildPid} =  
  Child:start_link(F, ..).
```

```
start_link(F, ...) ->  
  Env = wx:get_env(),  
  gen_server:start_link(  
    ..., [Env, F, ...]).  
  
init([Env, F, ...]) ->  
  wx:set_env(Env),  
  ...  
  {ok, #state{}}
```

# supervising widgets

```
start_link() ->  
  Env = wx:get_env(),  
  supervisor:start_link(  
    ?MODULE,[Env]).  
  
init([Env]) ->  
  {ok,{SupFlags,  
    #{start => {M,F,[Env]}}}}.
```

supervisor

```
start_link([Env,F,..]).  
  
init([Env,F,..]) ->  
  wx:set_env(Env),  
  ...  
  
wxButton:new(  
  F, ?wxID_ANY, [{label, "D"}])
```

child

# handling events a la wxWidgets

```
wxWindow:connect  
(  
  Widget,  
  EventType,  
  [{callback,  
    Callback}])
```



```
Callback(  
  Widget,  
  ..)
```

Callback invoked  
in a new process with  
the same environment

# additional way to handle them in wx

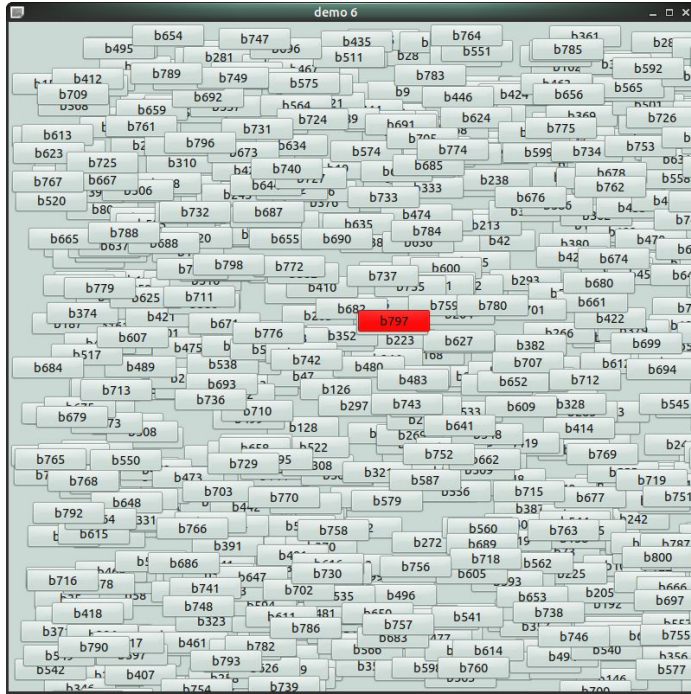
**wxWindow:connect**  
(  
    **Widget,**  
    **EventType)**

**Event sent to  
subscribing process as  
Erlang message:  
a #wx{} record**

**#wx{id, obj userData,event}**



# 1000 button frame demo



- 1000 supervised buttons
- subscribed to mouse click
- simple\_one\_for\_one supervisor
- click handled printing name and pid
- message *Fun/1* handled applying it to the button

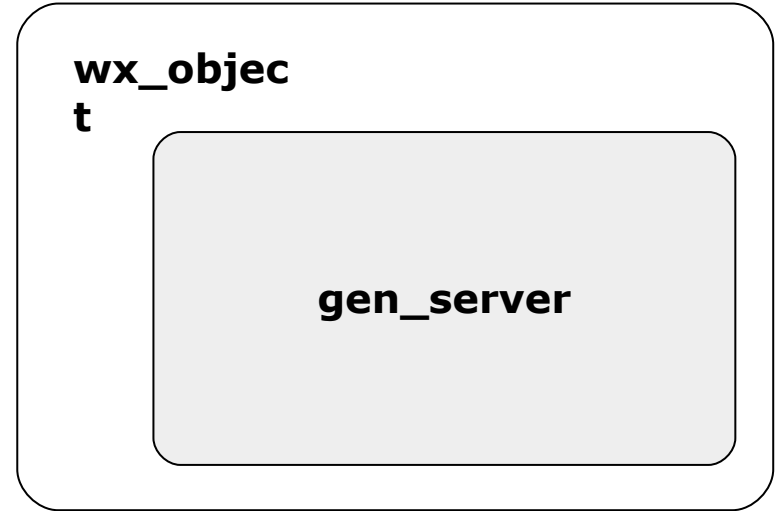
# destructor functions

- Widgets are created in a C++ constructor, not in Erlang
- We need to call destructor functions: `wxWindow:destroy/1`

```
terminate(_Reason, #state{widget = Widget}) ->  
  wxWindow:destroy(Widget),  
  ok.
```

# wx\_object behaviour

- handles import of wx environment from spawning process
- returns reference to widget
- forces handling of events sent as messages with `handle_event` callback
- provides a callback for synchronous event handling



# wx\_object creation

```
F = wxFrame:new(..),
```

```
Child =
```

```
Child:start_link(F, ...).
```

```
start_link(F, ...) ->
```

```
Env = wx:get_env(),
```

```
wx_object:start_link(  
  ..., [Env7 F, ...]).
```

```
init([Env7 F, ...]) ->
```

```
wx:set_env(Env),
```

```
...
```

```
{Widget, #state{}}
```

# asynchronous events handling

```
%% subscribe to event  
wxEvtHandler:connect(Widget, EventType)
```

```
%% handle event in wx_object  
handle_event(#wx{ }, State} ->  
...  
{noreply, State}
```

# synchronous events handling

```
%% subscribe to event  
wxEvtHandler::connect(Widget, EventType,  
[callback])
```

```
%% handle event in wx_object  
handle_sync_event(#wx{ }, Widget, State} ->  
...  
ok
```

undocum  
ented

# wx\_objects around us

wx:demo/0  
panels  
are wx\_objects

The screenshot shows a wxWidgets demo application. On the left is a code editor with the following code:

```
26 -include_lib("wx/include/wx.hrl").  
27  
28 -behaviour(wx_object).  
29 -export([start/1, init/1,  
30         terminate/2, code_change/3,  
31         handle_info/2, handle_call/3, handle_cast/2]).  
32  
33 -record(state,  
34         {  
35         parent,  
36         config  
37         }).  
38  
39 start(Config) ->  
40     wx_object:start_link(?MODULE, Config, []).  
41
```

On the right is a graphical window titled "Demo wxButton" containing a "Normal" button and a "Toggle" button. Below the buttons are controls for "Alignment Style" (Left Aligned, Top), "Other Styles" (Flat Style, Exact Fit), and "Stock Buttons" (About).

The screenshot shows the Erlang node monitor for a node named "nonode@nohost". The "Processes" tab is selected, displaying a table of running processes. The table has columns for Pid, Name or Initial Func, Reds, Memory, MsgQ, and Current Function.

Pid	Name or Initial Func	Reds	Memory	MsgQ	Current Function
<0.124.0>	wxe_server:init/1	43371	144960	0	gen_server:loop/7
<0.132.0>	erlang:apply/2	13117	75776	0	observer_pro_wx:table_holder/1
<0.83.0>	erlang:apply/2	2581	24564	0	timer:sleep/1
<0.131.0>	observer_pro_wx:init/1	2249	24800	0	wx_object:loop/6
<0.123.0>	observer	70	143224	0	wx_object:loop/6
<0.127.0>	timer_server	31	7148	0	gen_server:loop/7
<0.45.0>	application_master:init/4	2	4008	0	application_master:main_loop/2
<0.62.0>	supervisor_bridge:user_su...	2	2864	0	gen_server:loop/7
<0.66.0>	kernel_config:init/1	2	2820	0	gen_server:loop/7
<0.118.0>	wxe_server:init/1	2	35064	0	gen_server:loop/7
<0.126.0>	observer_sys_wx:init/1	2	184884	0	wx_object:loop/6
<0.128.0>	observer_perf_wx:init/1	2	11920	0	wx_object:loop/6
<0.129.0>	observer_alloc_wx:init/1	2	8904	0	wx_object:loop/6
<0.130.0>	observer_app_wx:init/1	2	7032	0	wx_object:loop/6
<0.134.0>	observer_port_wx:init/1	2	8904	0	wx_object:loop/6
<0.135.0>	observer_tv_wx:init/1	2	8948	0	wx_object:loop/6
<0.137.0>	observer_trace_wx:init/1	2	21704	0	wx_object:loop/6
<0.0.0>	init	1	0	0	init:wait_for_reque...
<0.1.0>	erts_code_purger	0	0	0	erts_code_purger:purge...
<0.2.0>	erts_literal_area_collector	0	0	0	erts_literal_area_collector:msg_L...

Number of Processes: 57

observer  
panels  
are wx\_objects



# terminating wx properly

**terminate(\_Reason, \_State) ->**

**wx:destroy(),**

**ok.**



# thank you!

```
QTD = wxMessageDialog:new (  
    wx:null(),  
    "Question Time",  
    [{style, ?wxICON_QUESTION}],  
  
wxDialog:showModal(QTD).
```

