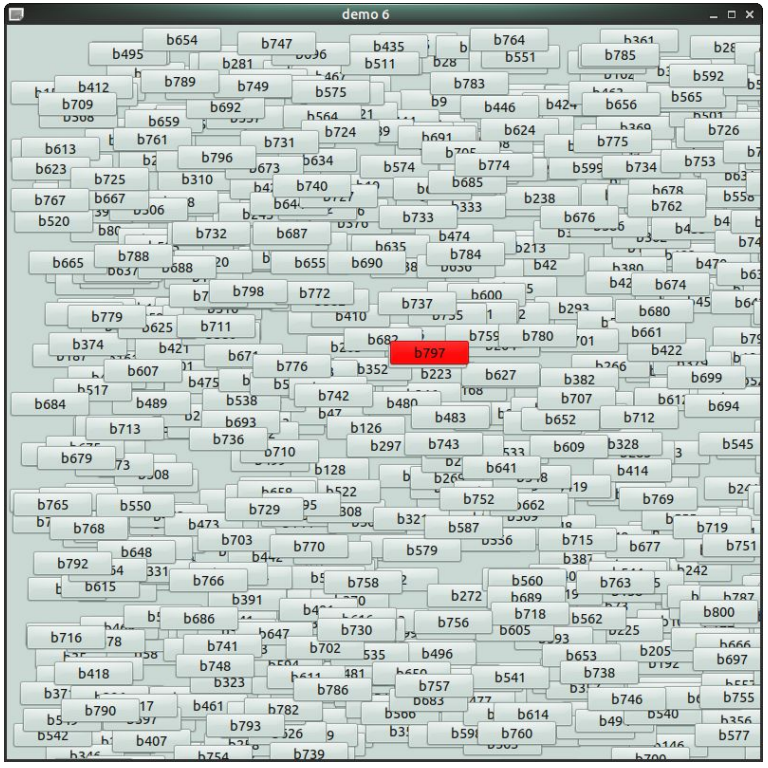
concurrency and wxErlang

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

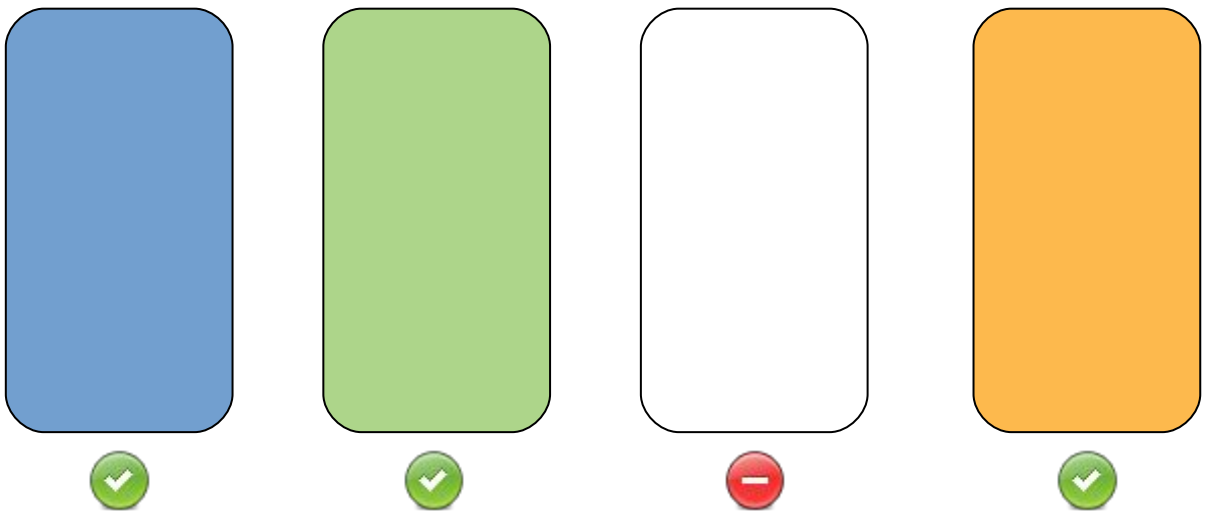
wxWidgets

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

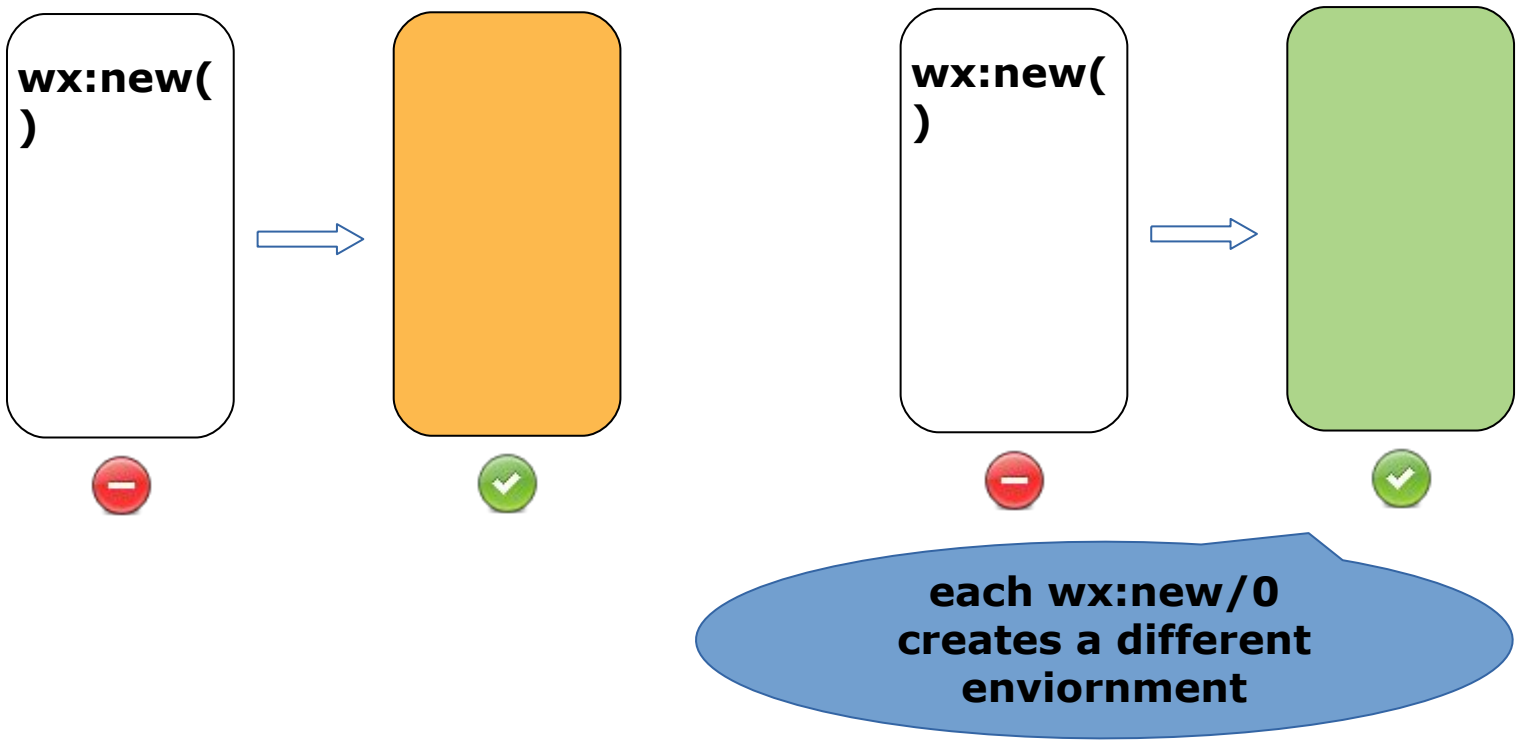
wxErlang

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

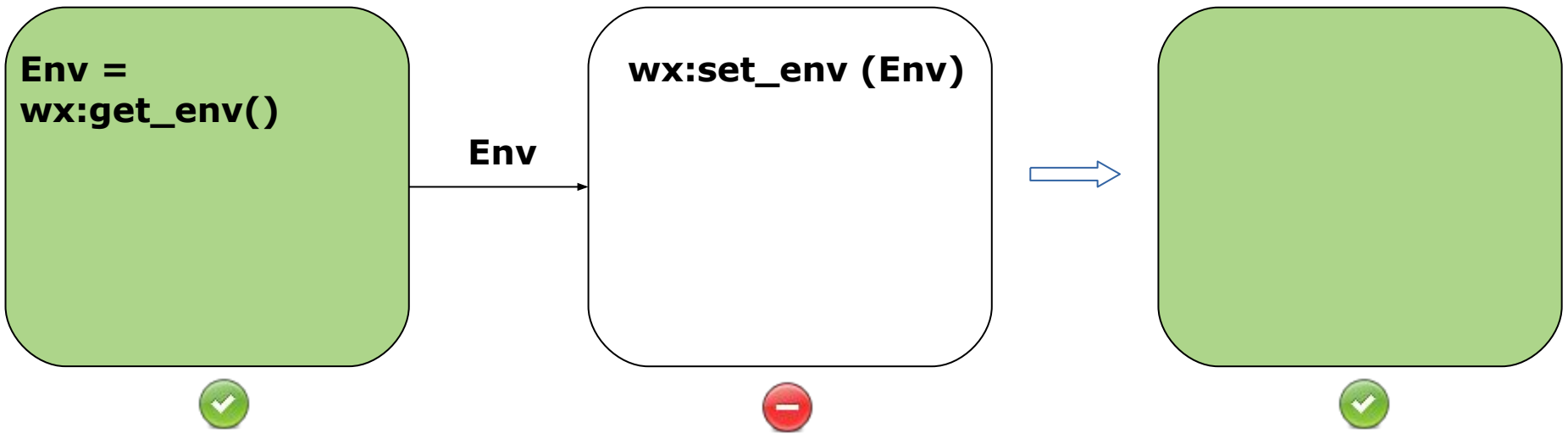
processes either have a wx environment or they don’t



either create the environment ...



or get it from some other process



why do we need the environment?

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

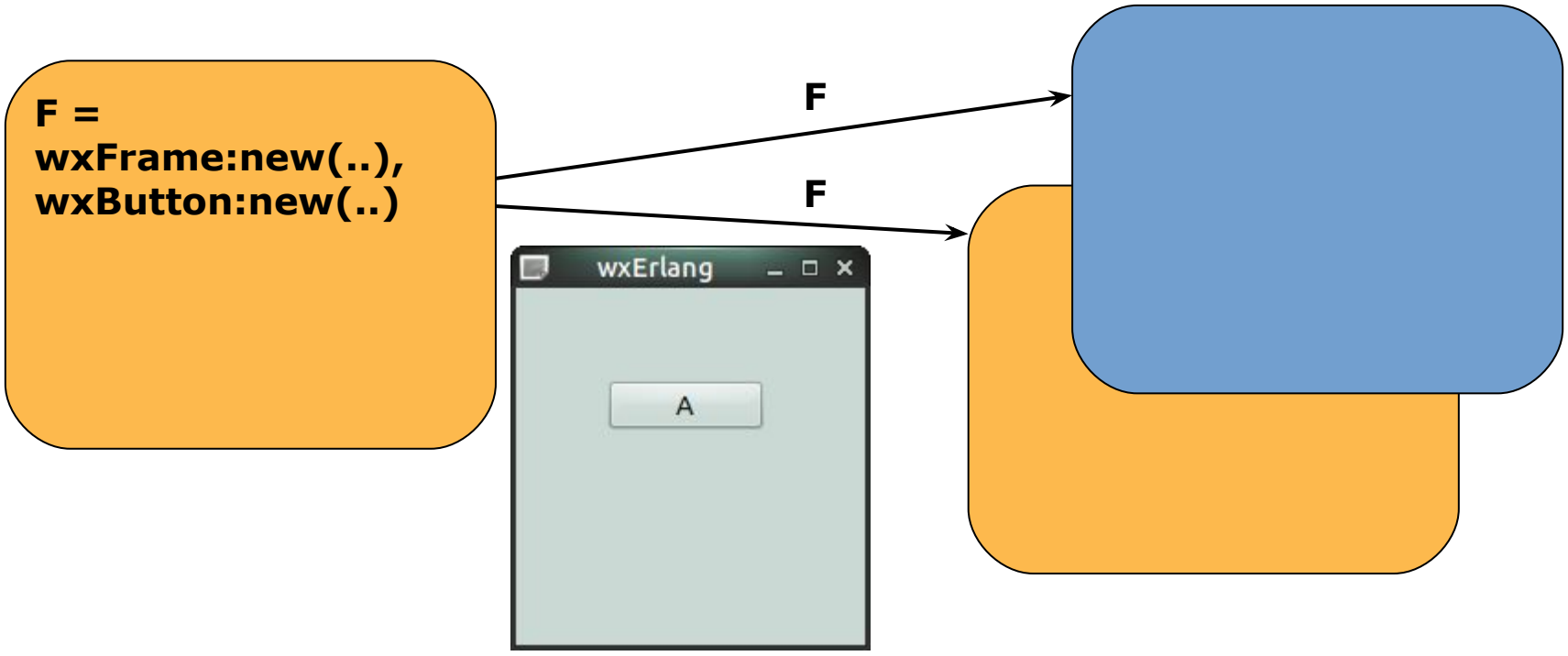

```
wxFrame:new(.
.)
```

children need a parent

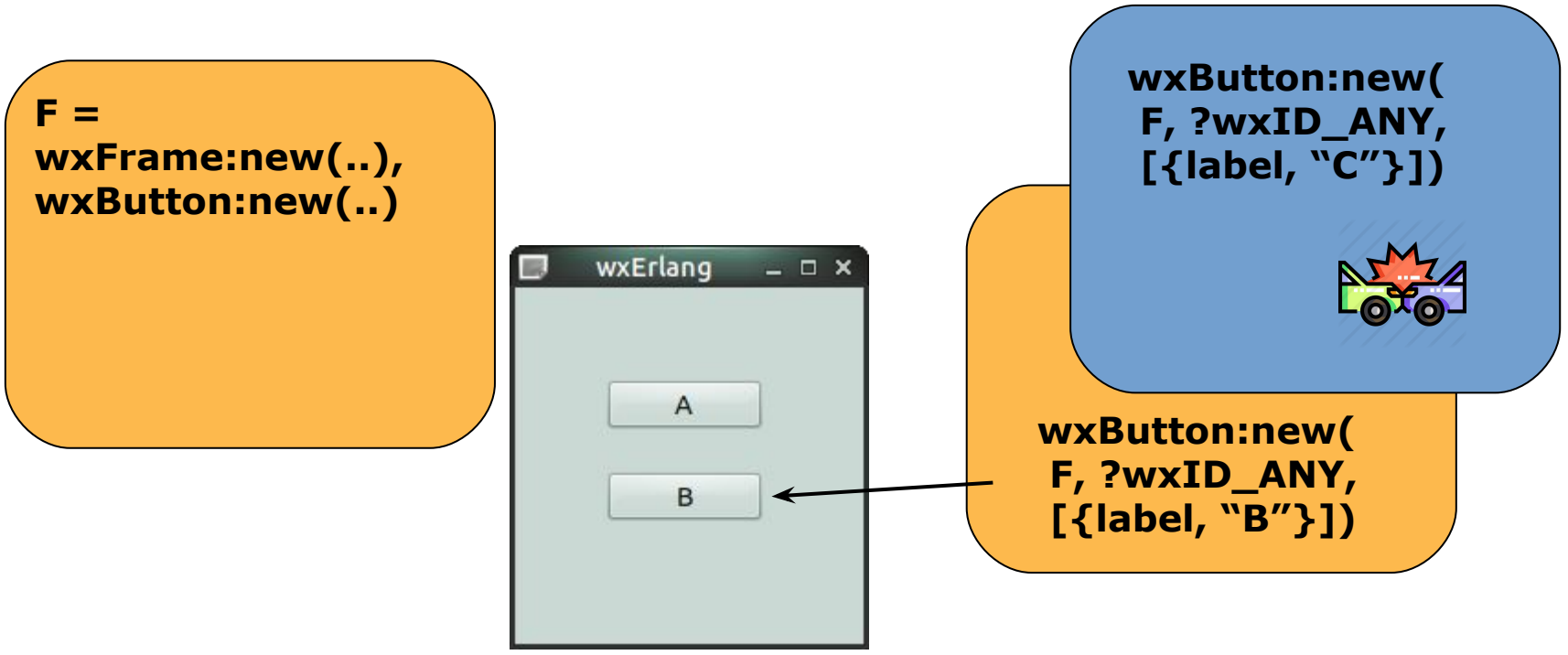
```
F =
wxFrame:new(..),

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



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

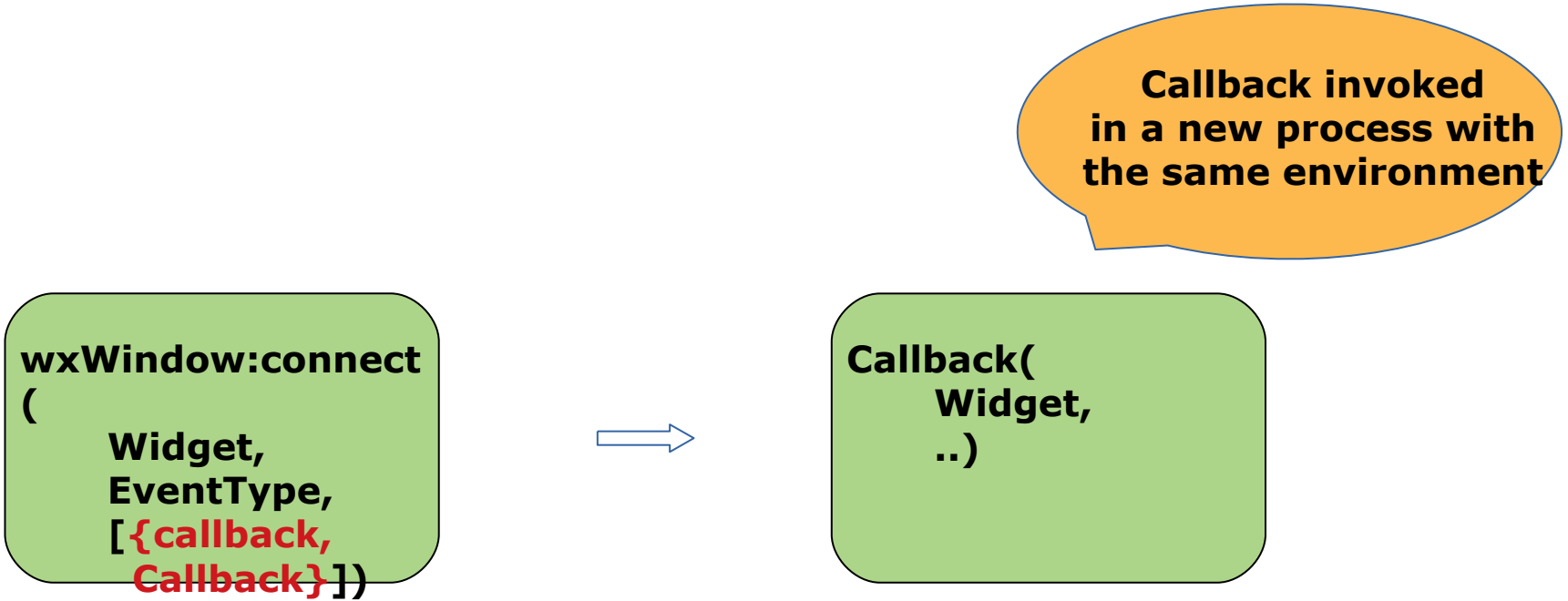
```
start_link() ->
 Env = wx:get_env(),
 supervisor:start_link(
     ?MODULE,[Env]).

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



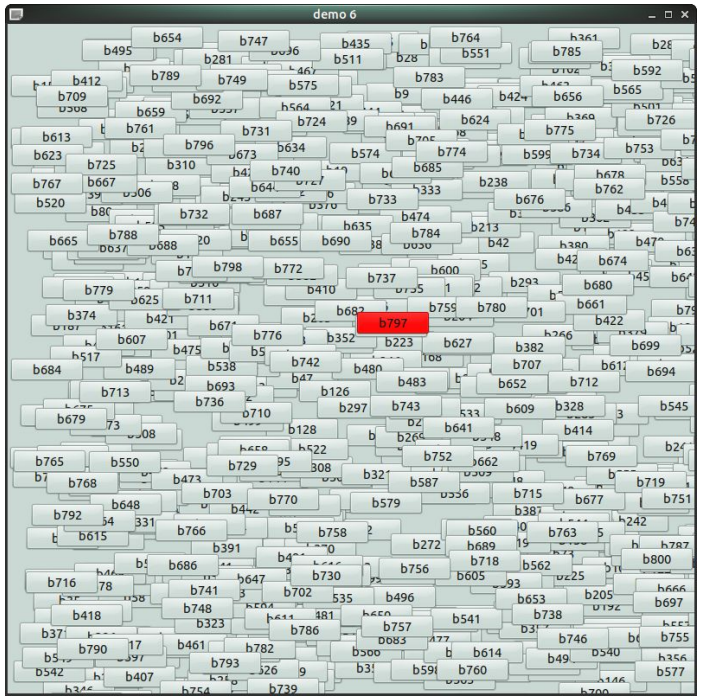
additional way to handle them in wx

```
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

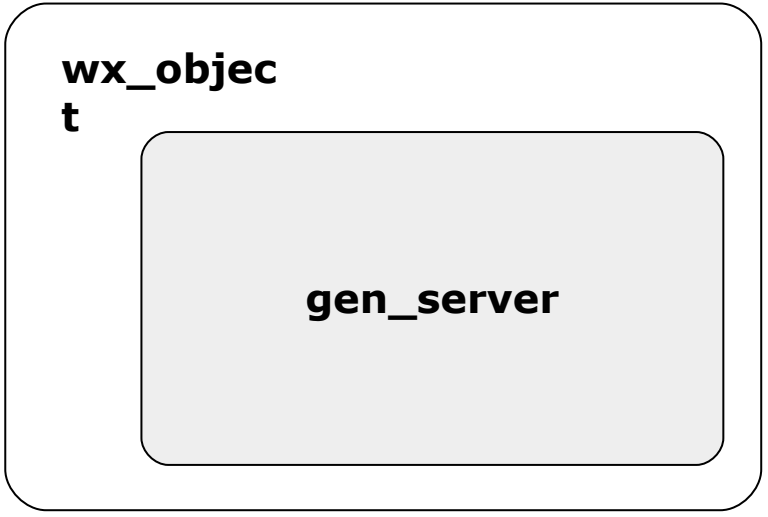
destructor functions

- 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(
      ..., [Env, F, ...]).

init([Env, 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}
```

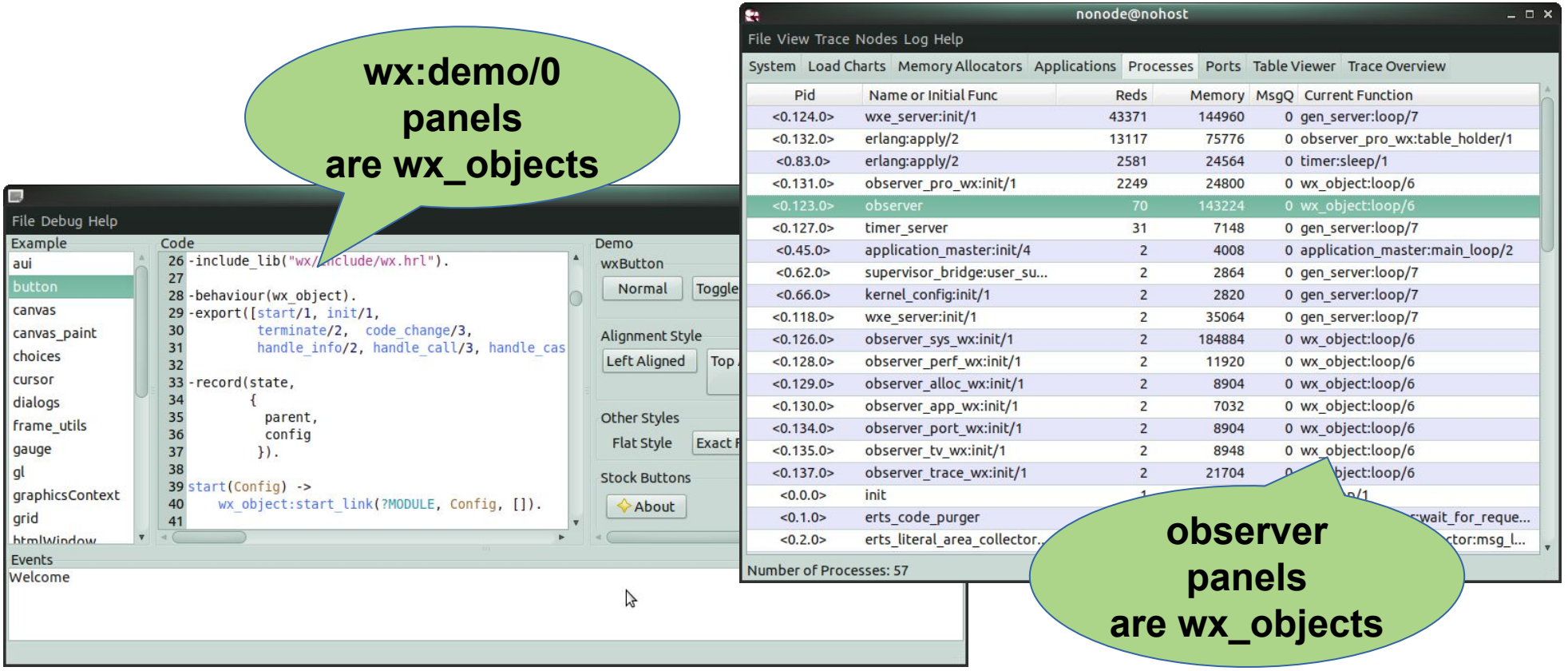
synchronous events handling

```
%% 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



terminating wx properly

```
terminate(_Reason, _State) ->
    wx:destroy(),
    ok.
```

thank you!

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

wxDialog:showModal(QTD).
```