Elixir is not Alone

Talking to other languages

XERPA

@nirev
Guilherme de Maio

who is

nirev?

@ São Paulo, Brasil

who is

nirev?

Working with Elixir since Sept 2015

XERPA

XERPA ?

XERPA

What?

A startup focused on solving HR bureaucracy in Brazil.

Which means: lots of system integrations,
lots of spreadsheets, lots of document storage,
lots of boring but ridiculously sensitive stuff

XERPA

Stack *(In Production)*

Elixir + Phoenix

ClojureScript + Reagent

PostgreSQL

ElasticSearch

Integrating with other languages

Why use other languages?

Elixir/Erlang

Why use other languages?

- Use the right tool for the job.
  Elixir/Erlang is great, but not for everything

- Maybe you don’t have the time.
  It takes time to implement something. What if you can’t invest time reimplementing something that is already there in other language?

XERPA

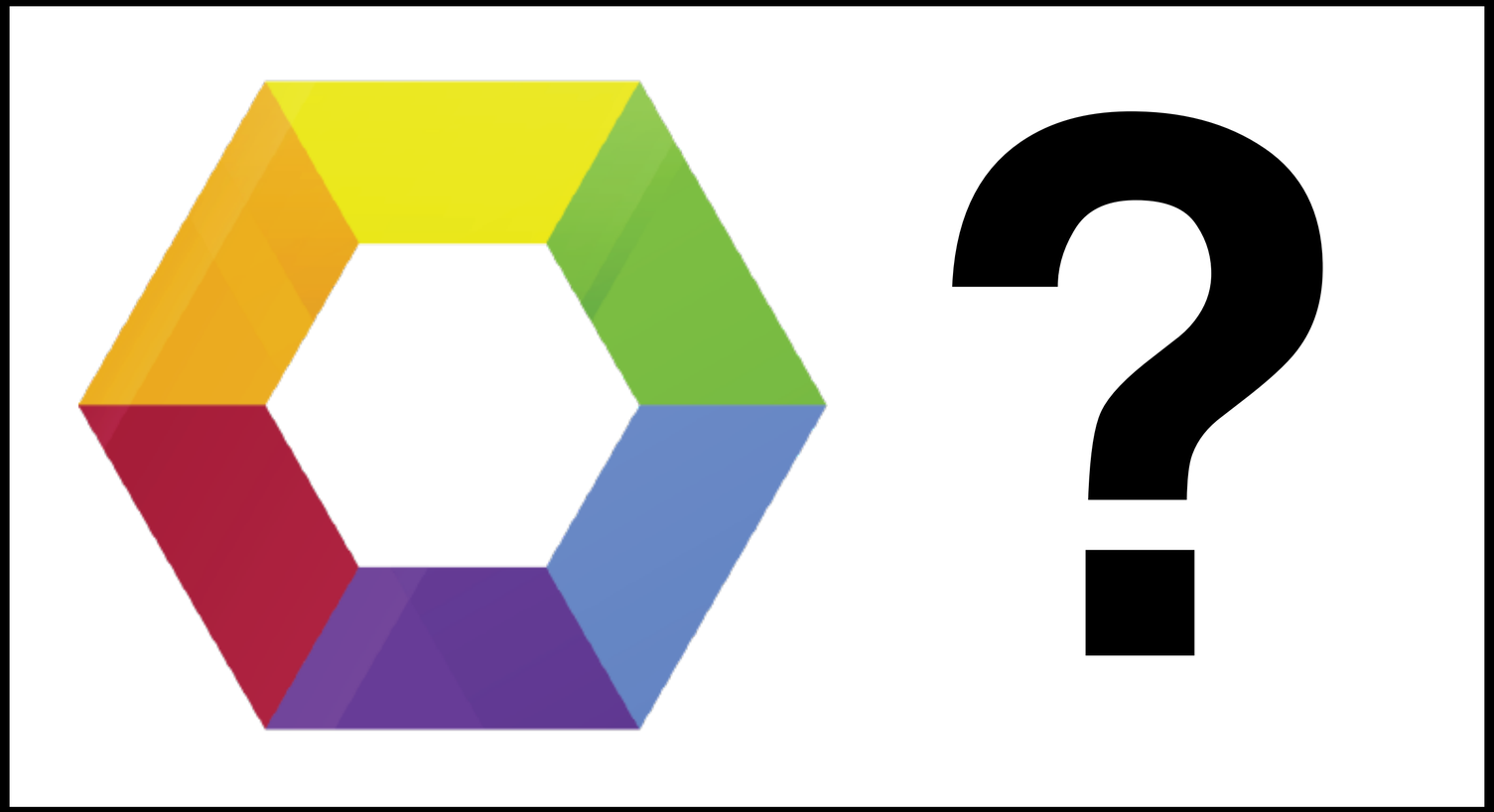
Features

Image Processing

Import/Export
spreadsheets
(docx; xlsx)

XERPA

Features



XERPA

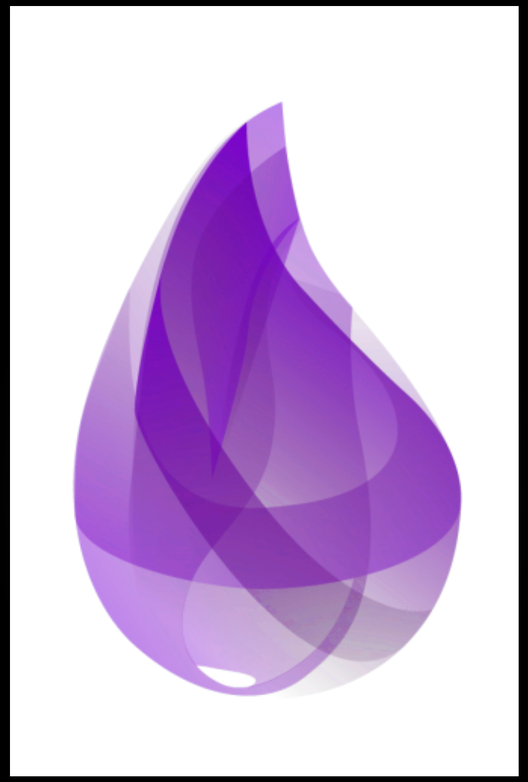
Features

XERPA

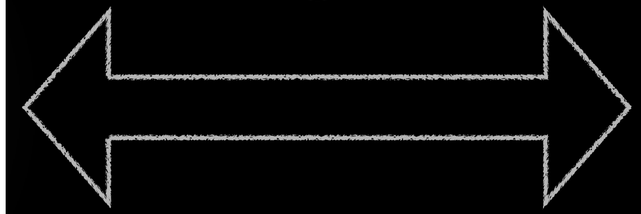
Features

Image Processing

Import/Export spreadsheets (docx; xlsx)



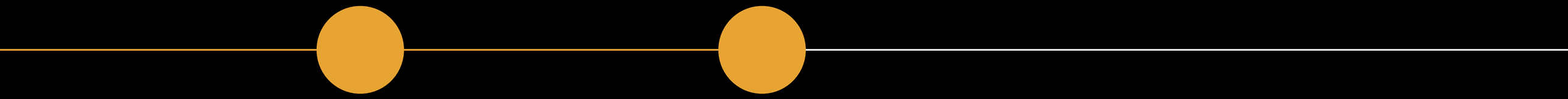
?




GraphicsMagick


OPENXML

How to integrate other languages to your Elixir/ Erlang codebase?



Elixir/Erlang

Interoperability Options

Ports

NIFs

Thrift

Port Drivers

APIs

Nodes

Elixir/Erlang

Ports

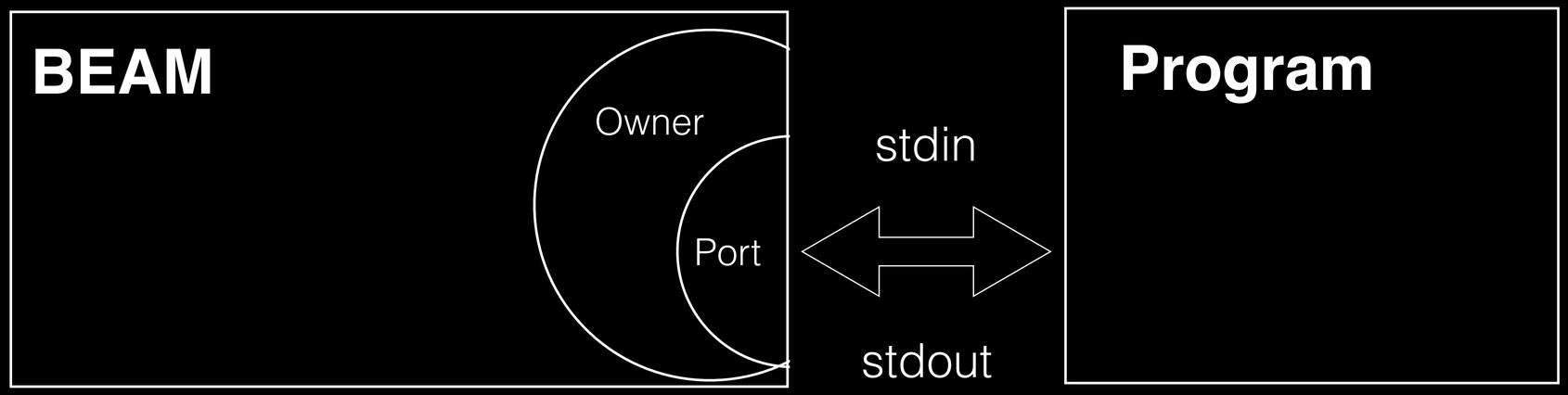
Elixir/Erlang

Ports

- THE standard way to communicate with the Otherworld, outside of the BEAM
- It’s STDIN/STDOUT bridge to other programs which reside in another OS process.
- Each port is owned by a single Erlang process, and only that process can talk to the port. If the process ends, so does the port.
- Elixir’s System.cmd uses Ports, for example.

Elixir/Erlang

Ports



IT's **safe**:

- When the program dies/crashes, only the port dies
- When the owner dies, so does the port and pipes are closed

Elixir/Erlang

Ports: caveats

- Programs that wait till EOF to emit output: when closing a port, you close both pipes. There's no way to receive after. (alternative: Porcelain, DIY wrapper, other libs?)
- Communication is streamed. No guarantees of chunks sent/received together. So parse it, char by char!
- No specific encoding format. So encode as you like: Erlang Term Format, JSON, bytes, etc..
- Zombie processes

Elixir/Erlang

Ports

```
iex(1)> port = Port.open({:spawn, "cat -n"}, [:binary])
#Port<0.1169>
iex(2)> Port.command(port, " hello\n")
true
iex(3)> send port, {self(), {:command, " hello, again!\n"}}
{#PID<0.80.0>, {:command, " hello, again!\n"}}
iex(4)> flush()
{#Port<0.1169>, {:data, "     1\t hello\n"}}
{#Port<0.1169>, {:data, "     2\t hello, again!\n"}}
:ok
```

Elixir/Erlang

NIFs: Native Implemented Functions

Elixir/Erlang

NIFs: Native Implemented Functions

Is a way to implement code in C (or a language compatible) that is loaded as shared libraries by the BEAM

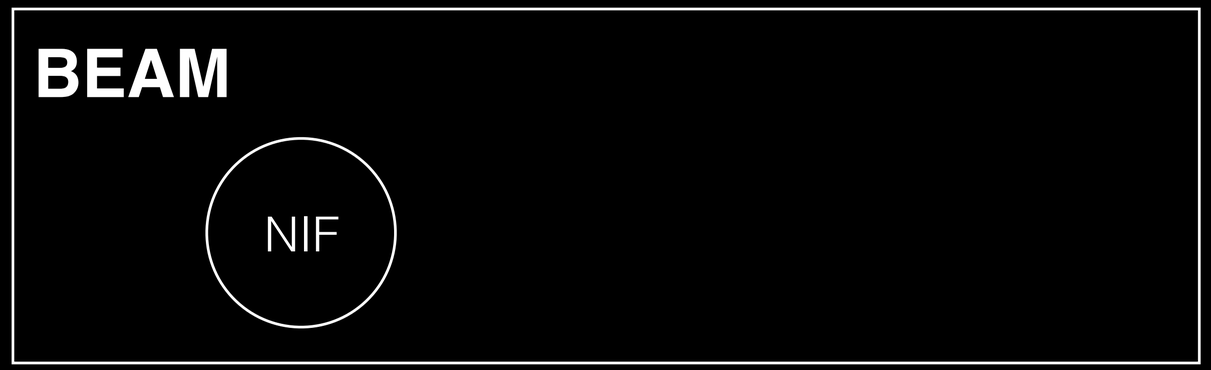
Code is exposed as functions of a module in Elixir/Erlang for those calling it

Simpler than ports in some aspects: no need to encode data, and no need to use STDIN, STDOUT

It’s faster.

Elixir/Erlang

NIFs: Native Implemented Functions

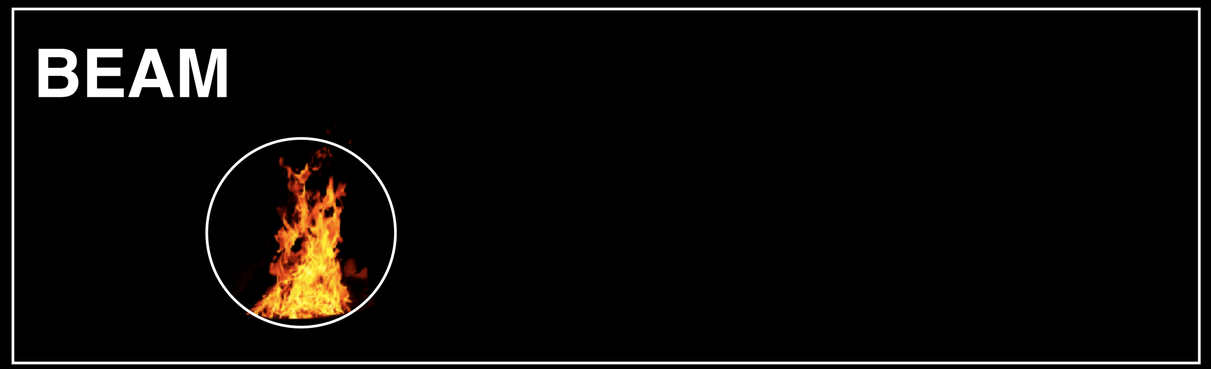


A NIF is executed as a direct extension of the VM.
Meaning: it’s not done in a safe environment.

The VM can’t provide same guarantees when executing Erlang/Elixir code: no preemptive scheduling or memory safety.

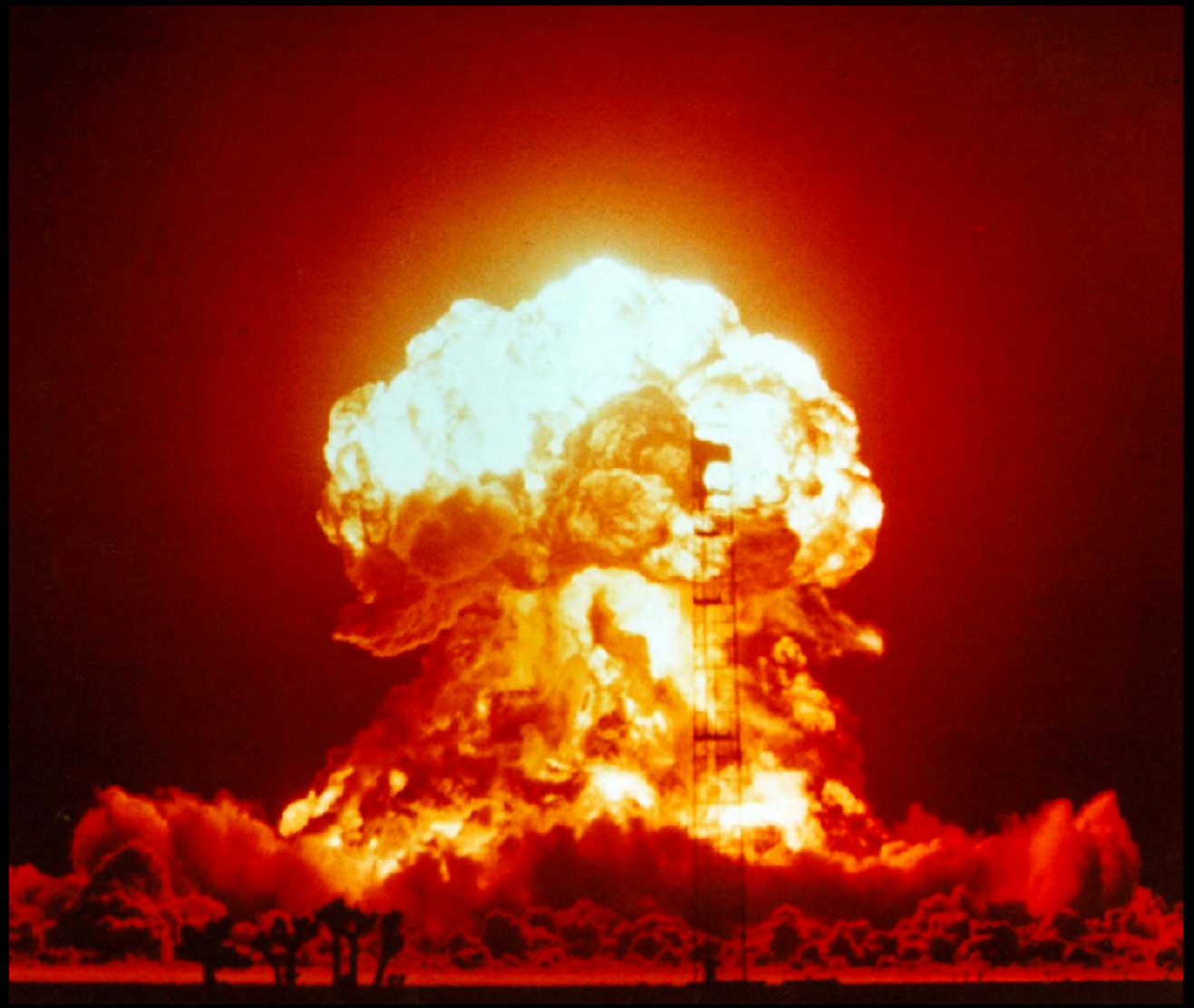
Elixir/Erlang

NIFs: Native Implemented Functions



Elixir/Erlang

NIFs: Native Implemented Functions



Elixir/Erlang

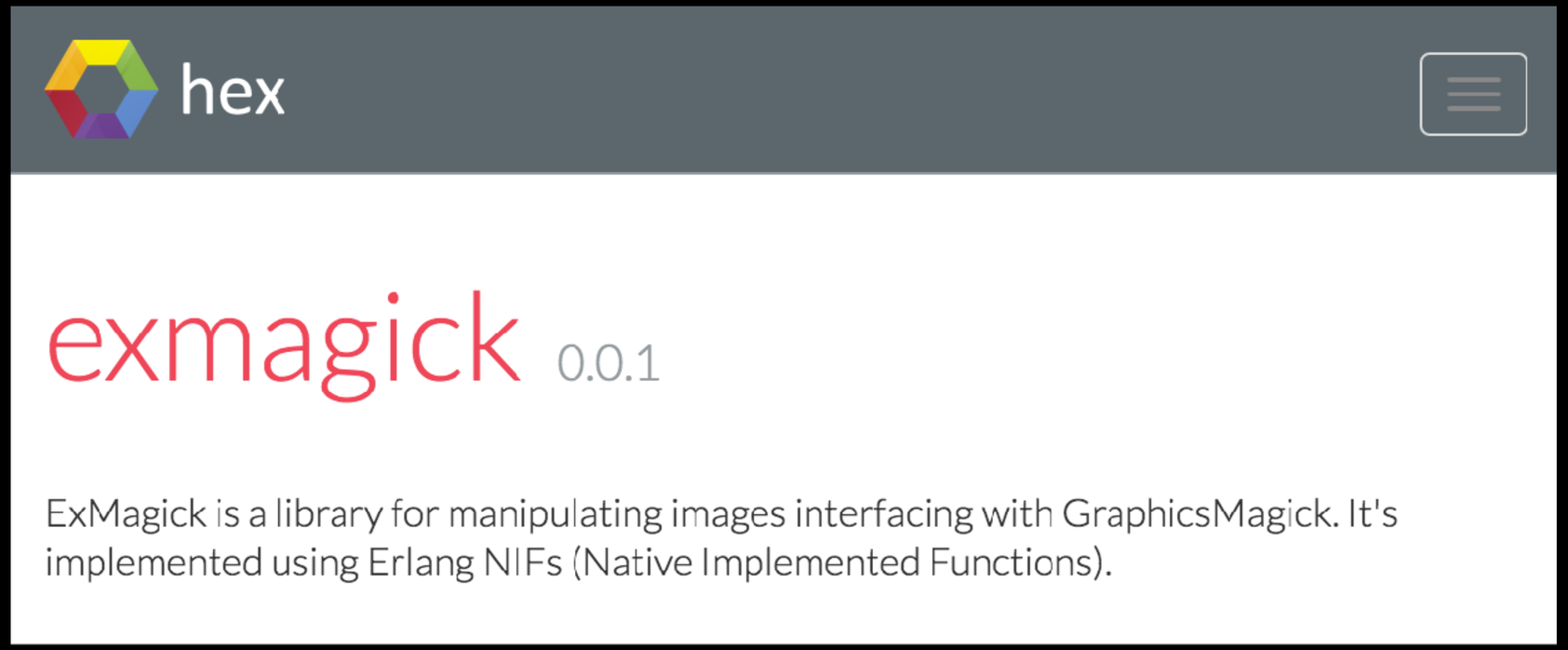
NIFs: don’t be afraid

Although it’s less safe, don’t be afraid of using it:

- Several libs are implemented with NIFs. Markdown parser for example
- Dirty Schedulers are enable by default in newer Erlang releases
- Rustler: safer NIFs implemented with Rust :)

Elixir/Erlang

NIFs: Examples



Elixir/Erlang

NIFs: Examples

```c
#include "erl_nif.h"

ERL_NIF_TERM sum(ErlNifEnv* env, int argc, const ERL_NIF_TERM argv[])
{
  int a, b;
  enif_get_int(env, argv[0], &a);
  enif_get_int(env, argv[1], &b);
  return enif_make_int(env, a + b);
}

static ErlNifFunc nif_funcs[] = {
  {"sum", 2, sum}
};

ERL_NIF_INIT(Elixir.NifSum, nif_funcs, NULL, NULL, NULL, NULL);
```

Elixir/Erlang

NIFs: Examples

```elixir
defmodule NifSum do

  @on_load :init
  def init do
    :ok = :erlang.load_nif('./priv/nif', 0)
  end

  def sum(_, _), do: {:error, :nif_library_not_loaded}
end
```

Elixir/Erlang

Port Drivers

Elixir/Erlang

Port Drivers

It’s kind of a mix between NIF and Port.

You create a port, but for a process living inside the BEAM.

Like NIF:
- it’s loaded as a share library (.so)
- there’s no context switch
- if it breaks, it breaks it all

The main difference is: you’re implementing an Erlang process in C, as so it can by async and react to events/ messages!

(but it’s harder to implement)

Elixir/Erlang

Thrift

Elixir/Erlang

Thrift

Apache Thrift is an RPC framework created by Facebook. Kinda like the “the sucessor of CORBA”

It provides an Interface Definition Language, to create data types and function signatures that can be implemented in a lot of languages.

For Elixir, there is Pinterest’s riffed

Supports: java, c, c++, python, ruby, Haskell, perl, php, and more

Serialization with binary format, quite fast

Elixir/Erlang

Nodes

Elixir/Erlang

C/Java Nodes

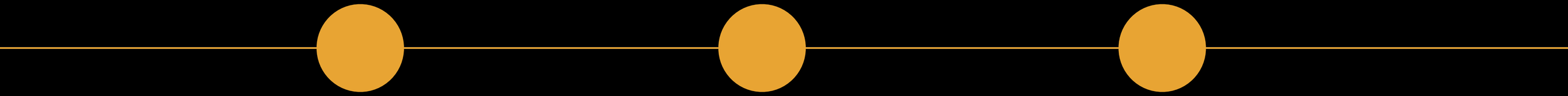
Using Erl_Interface in C or Jinterface in Java.

Those libraries make possible for you to run a C/Java program that behaves like a distributed Erlang node.

It’s not coupled with your app, and it’s possible to detect failures in the remote node.

IMO, makes more sense when it’s an application that can co-exist but not necessarily depend of one another.

So, what do we get from all of this?



Takeaways

- There are a lot of ways to integrate
- Consider Performance vs Safety
- Choose what is best for your case
- In doubt, go the easy and safer way.
  Optimize later ;)


Elixir/Erlang is not an island

Elixir/Erlang

References

Links for everyone11!!

- http://erlang.org/doc/tutorial/introduction.html
- http://erlang.org/doc/man/erl_nif.html
- http://theerlangelist.com/article/outside_elixir
- https://github.com/knewter/complex
- https://github.com/alco/porcelain
- http://elixir-lang.org/docs/stable/elixir/Port.html
- https://github.com/Xerpa/exmagick
- https://github.com/hansihe/rustler
- https://github.com/pinterest/riffed
- https://hackernoon.com/calling-python-from-elixir-erlport-vs-thrift

Thank you!

XERPA

@nirev
Guilherme de Maio