Enter the Erlang 🐲🐉 with LFE

by David Cao

What is Erlang?

Dracula

What is a Erlang ?
A miserable little
pile of secrets.

Why Erlang?

- It is a technology tested for 3️⃣0️⃣+ years on the industry ☎️⟶💻⟶📱
- It manage concurrency in a healthy way ❕
- Soft-real time server side
- Fault Tolerance
- OTP: real design patterns ~~(Gang of Four)~~
- The VM is more an OS rather than an bytecode's interpreter del

You don't have to know OTP ❕

If you already know a little of lisp, you can learn the ecosystem on the go 🙇📚

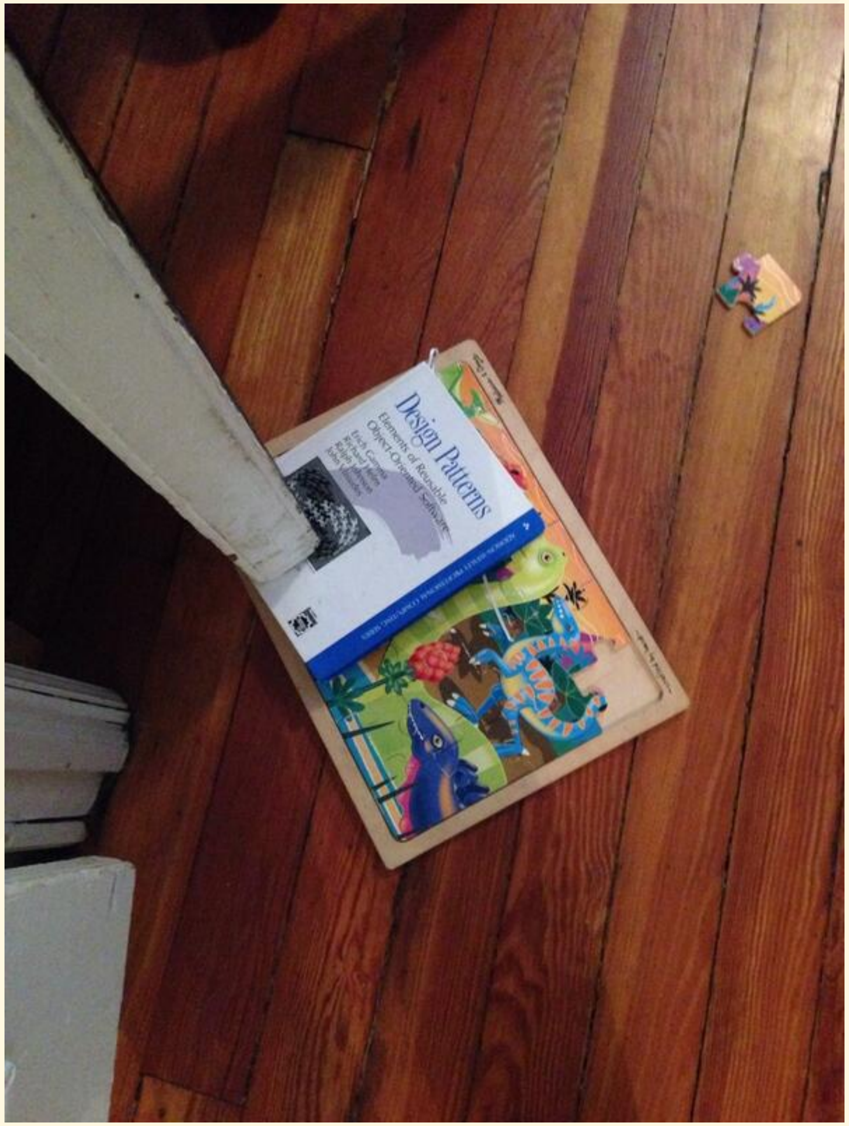
Erlang is good 💯❗ for soft real time ⏰

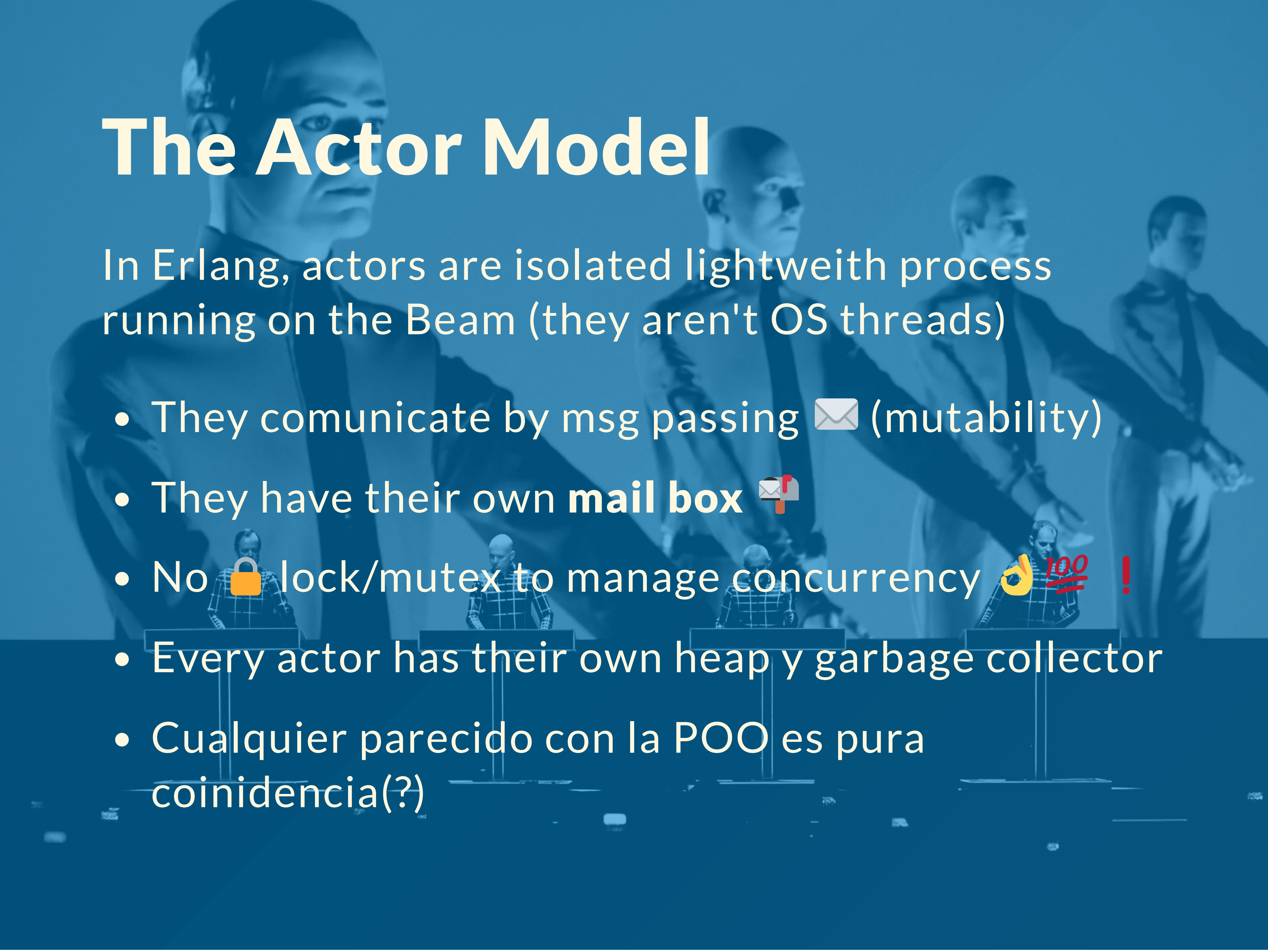
Real ⏰ -> missing a deadline is a total system failure.

Soft 🍨 Real ⏰ ->the usefulness of a result degrades after its deadline📉, but it still usefull et al ❗.
In streaming is valued the continuity of service.

The most of te services nowadays are soft-real time 24/7

OTP: Real Life Design Patterns





The Actor Model

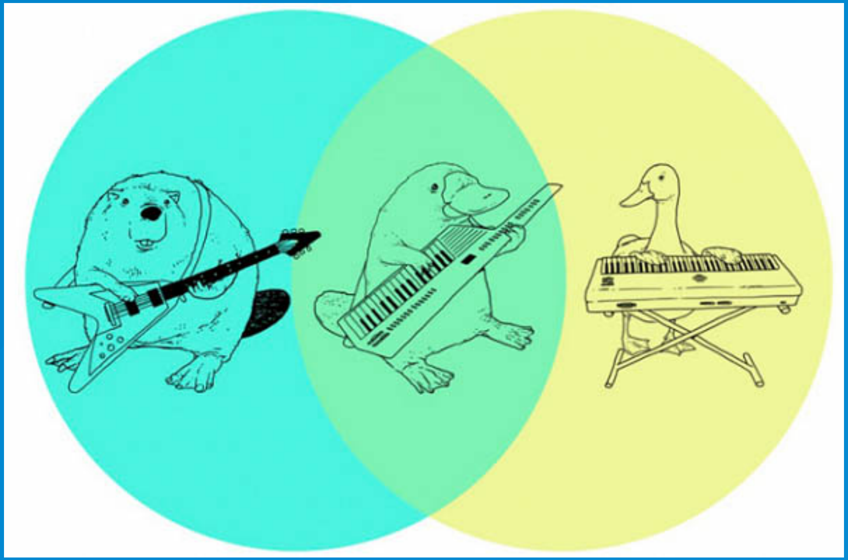
In Erlang, actors are isolated lightweith process running on the Beam (they aren't OS threads)

- They comunicate by msg passing ✉️ (mutability)
- They have their own **mail box** 📬
- No 🔒 lock/mutex to manage concurrency 👌💯 ❗
- Every actor has their own heap y garbage collector
- Cualquier parecido con la POO es pura coinidencia(?)

Behaviour

It's A pattern desing 😄, es parecido a la herencia en POO o una Java's interface.

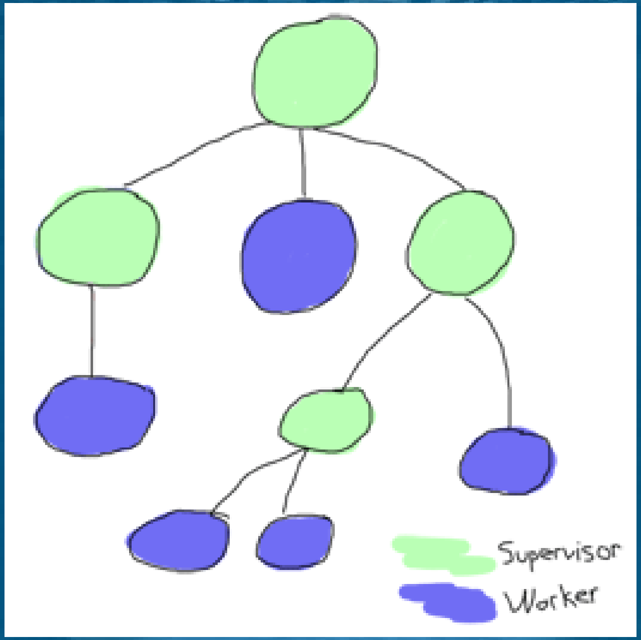
An amount of callbacks (contract) must be defined for the **behaviour** works.



Supervisor

supervisor is an **behaviour** responsable de arrancar, parar y monitorear sus procesos hijos.

Siempre es un nodo del arbol de procesos, en cambio los procesos worker son hoja o terminales.



Gen Server (aka Microservices)

It is a pattern for code generic server in Erlang.
La idea es separa la funcionalidad del manejo de la concurrencia del servidor a travez de callbacks.

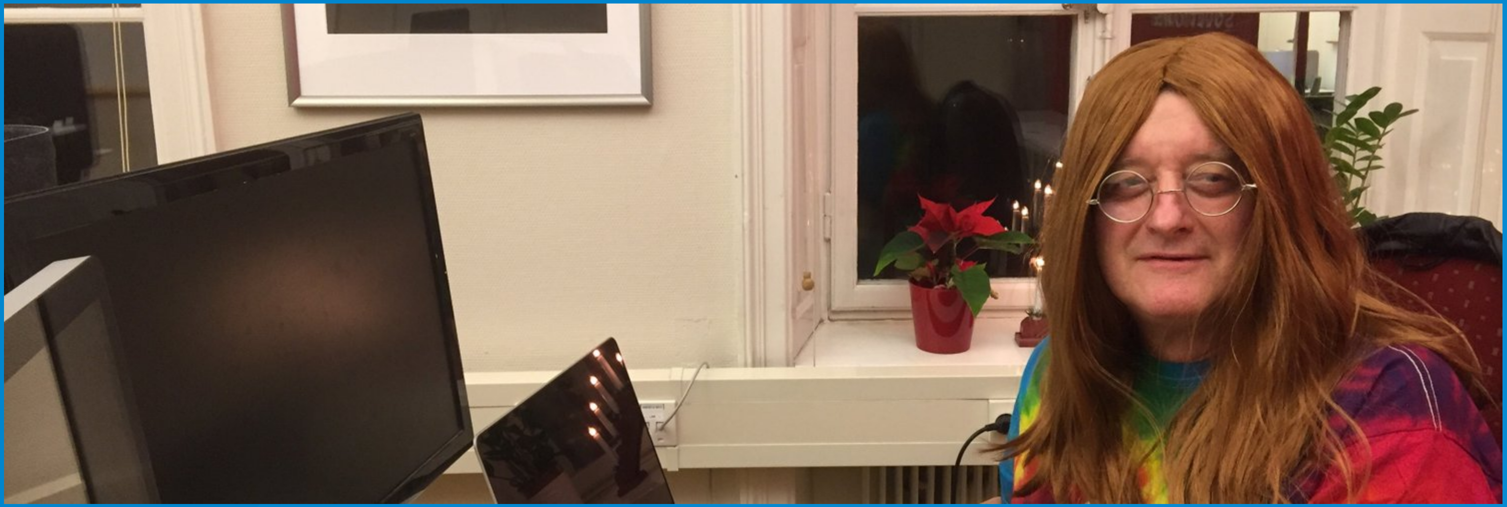
handle_call: synchronou calls (wait for an answer)
handle_cast: asynchronou calls (no wait for an answer)

the code inside the handler will be served by a Beam's process.

`(lisp (flavoured (erlang)))`

LFE es LISP's dialect created by
Roberto Virding sobre la Erlang VM.

Es un Lisp2+, LFE tiene diferentes namespaces.
Podes tener una fun `help` y una var `help`



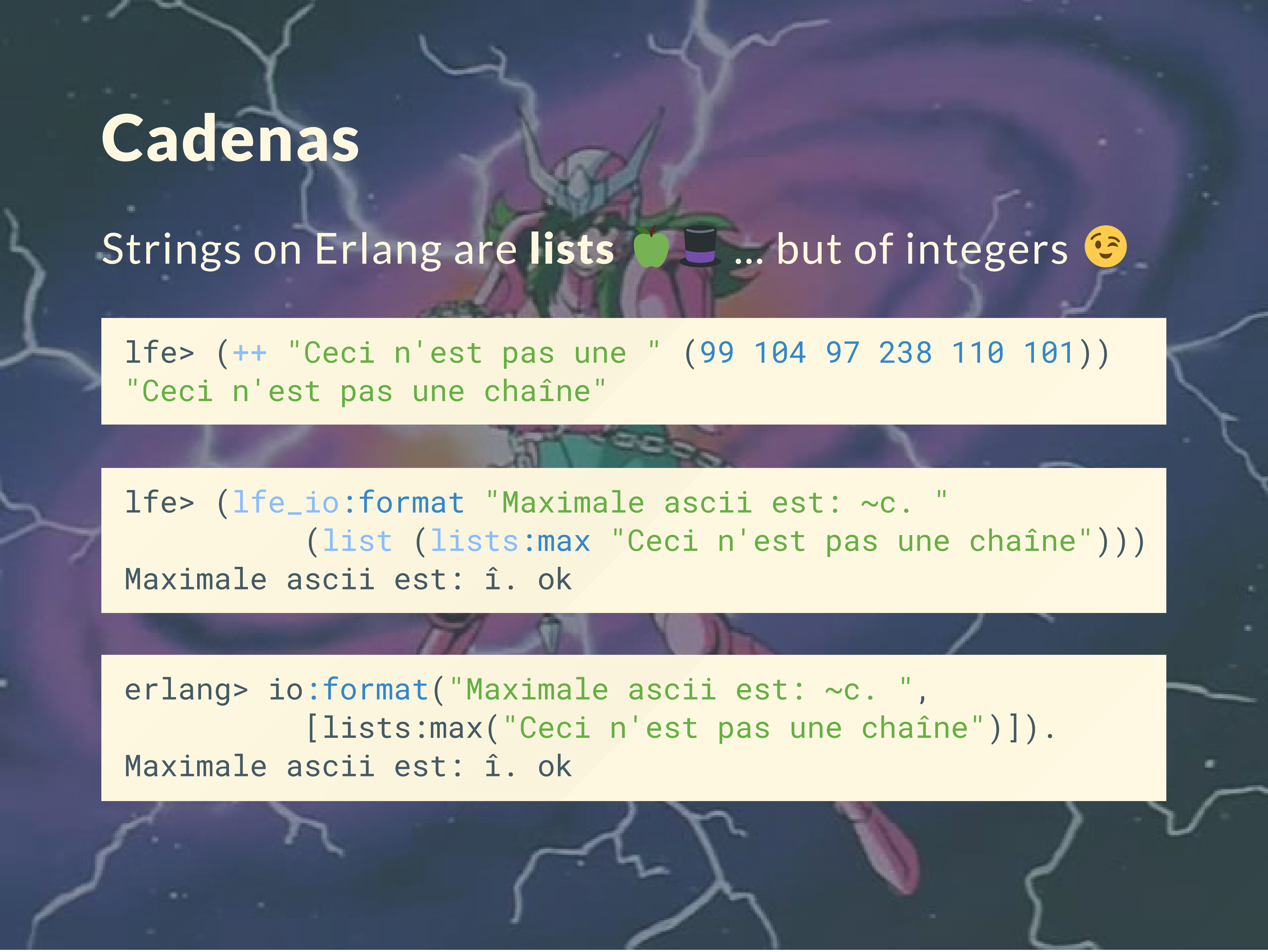
Numbers

Integers could so big that you want or you get out of memory, whichever occurs first. 😅

```
lfe> ( + 1000000000  99999999999999999999999999999999)
100000000000000000000000000999999999
```

There is not '**nan** or '**infinity**, you have to create it.

```
lfe> ( /  1000000444  991)
1009082.1836528758
lfe> (/ 1.0  0.0)
exception error: error in arithmetic expression
  in (erlang : / 1.0 0.0)
```



Cadenas

Strings on Erlang are **lists** 🍏🎩 ... but of integers 😉

```
lfe> (++ "Ceci n'est pas une " (99 104 97 238 110 101))
"Ceci n'est pas une chaîne"
```

```
lfe> (lfe_io:format "Maximale ascii est: ~c. "
          (list (lists:max "Ceci n'est pas une chaîne")))
Maximale ascii est: î. ok
```

```
erlang> io:format("Maximale ascii est: ~c. ",
          [lists:max("Ceci n'est pas une chaîne")]).
Maximale ascii est: î. ok
```

Atoms

They are **enums** que se representan itself, also atoms starts with con comilla simple **'**.

```
lfe> (erlang:is_atom 'atom)
true
```

```
erlang> erlang:is_atom(atom).
true
```

- **'true** has a `truthy` valuey and **'false** `falsy`
- No hay **null**🚫, pero podĂŠs definir el ĂĄtomo **'null** , **'undefined**, **'none**, **'nothing**, **'lol** , **'ahre**

Tuplas

```
lfe> (tuple 'ok "I am a pickle!")
#(ok "I am a pickle!")
lfe> (tuple 1 2 3 4 5)
#(1 2 3 4 5)
lfe> #(1 9)
#(1 9)
```

Maps 🌎

```
lfe> (map 'key 'value)
#M(key value)
lfe> (map 'lfe  "Erlang" 'creator "Robert Virding")
#M(lfe "Erlang" creator "Robert Virding")
```

Modules, Functions and Pattern Matching

```
lfe> (set (tuple 'error msg) (tuple 'error "Error :("))
#(error "Error :(")
lfe> msg
"Error :("
```

Every module name is an **atom()!**
Also el Pattern Matching es conceptualmente similar to OOP's dispacher.

```
 (defmodule conversion
  (export (convert-length 1)))

(defun convert-length
  (((tuple 'centimeter x)) (tuple 'inch (/ x 2.54)))
  (((tuple 'inch y)) (tuple 'centimeter (* y 2.54))))
```

Modules, applicacions and libraries

En Erlang tener un **main(args ...)** no tiene mucho sentido (como punto de entrada), es mas para nostalgicos de otros lenguajes.

Dado que que podes tener varios procesos escuchando.

Exersism's Example

```
(defmodule leap
  (export all))

(defun leap-year
  ((year)  (when (== 0 (rem year 400)))
    'true)
  ((year)  (when (== 0 (rem year 100)))
    'false)
  ((year)  (when (== 0 (rem year 4)))
    'true)
  ((_year)
    'false))
```

Exersism's Example

```
(defmodule rna-transcription
  (export (to-rna 1)))

(defun to-rna-char
    ([#\G] #\C)
    ([#\C] #\G)
    ([#\T] #\A)
    ([#\A] #\U))

(defun to-rna (dnaList)
    (lists:map #'to-rna-char/1 dnaList))
```

Erlang came with 3 DBs build-in

ETS (Erlang Term Storage) is a inmemory BD that vcan save any kind of erlang term()

DETS (Disk ETS) es similar a la ETS pero con persistencia en disco con un lĂmite de 2G.

Mnesia es una capa comstruida sobre la ETS y la DETS que permite transaciones.

The most used are ETS y Mnesia

DOCK 3
17
17
2500
25
CANEJO

Dieselpunk