Erlang Factory 2015
A brief resume

(because it was not only about aki)

Amilcar

What is this conference about?

The Erlang Factory is a two-day tech conference focused on the Erlang programming language.

Where was it?

San Francisco Bay Area, California

Amilcar wants to ...

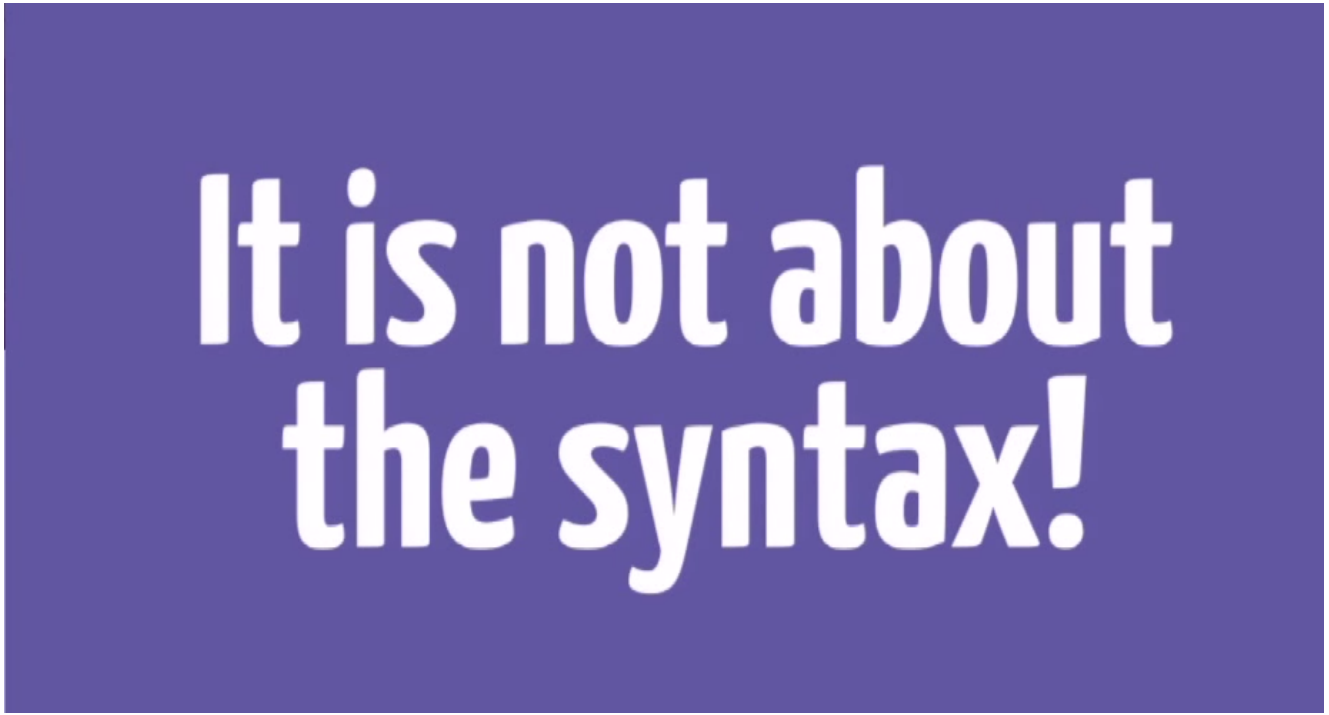
Talk a little bit about one of the presentations.

Jose Valim - “What Elixir is about”

What Elixir is about

“Elixir is a dynamic, functional language designed for building scalable and maintainable applications.”

What Elixir is about



Goals

elixir

- Extensibility
- Productivity
- Compatibility

Extensibility

elixir

PROTOCOLS

Protocols are a mechanism to achieve polymorphism in Elixir.

Extensibility

elixir

PROTOCOLS + STRUCTS

THE POSTA

Productivity

elixir

Tooling

Mix + Hex + Docs

Productivity

Mix is a build tool that provides tasks for creating, compiling, testing (and soon releasing) **Elixir** projects

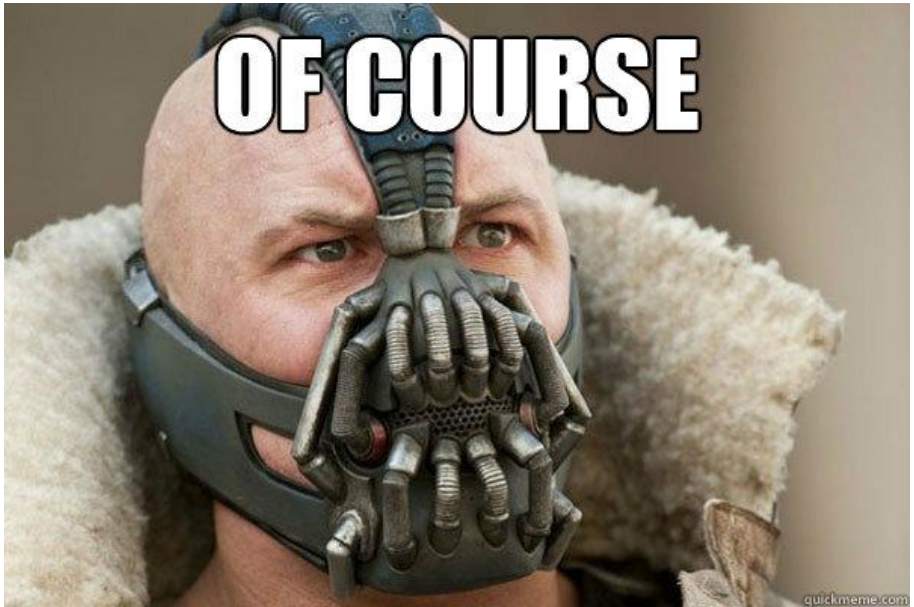
Productivity

elixir

```
$ mix new foo
$ cd foo
$ mix test
$ mix hex.publish
$ mix hex.docs
```

Compatibility

Elixir is 100% Erlang compatible



Cool things

- Pipe lines
- Agents
- Tasks (with supervisors)
- Macros (?)
- and more ...

Do you want more?

Visit us…

Next Friday I will give a (little bit longer) talk about Elixir

Hernan

Kenji Rikitake's "Searching for Better PRNGs"

Summary:

Do not try inventing your own random number generators!

Summary part 2:

And don’t use erlang’s default PRNG either!

PRNGs are everywhere

- Rolling dice (for games)
- (Property) testing (QuickCheck, ProPer, Triq)
- Variation analysis of electronic circuits
- Network congestion and delay analysis
- Risk analysis of project schedules
- Passwords (Secure PRNGs only!)

How PRNG work

- Sequential iterative process
- For multiple processes, seeds and other parameters should be chosen carefully to prevent sequence overlapping

NOT in this talk: Secure PRNGs

- For password and cryptographic key generation with strong security
- Use crypto:strong_rand_bytes/1
- Remember entropy gathering takes time
- This is cryptography - use and only use proven algorithms! Do not invent yours!

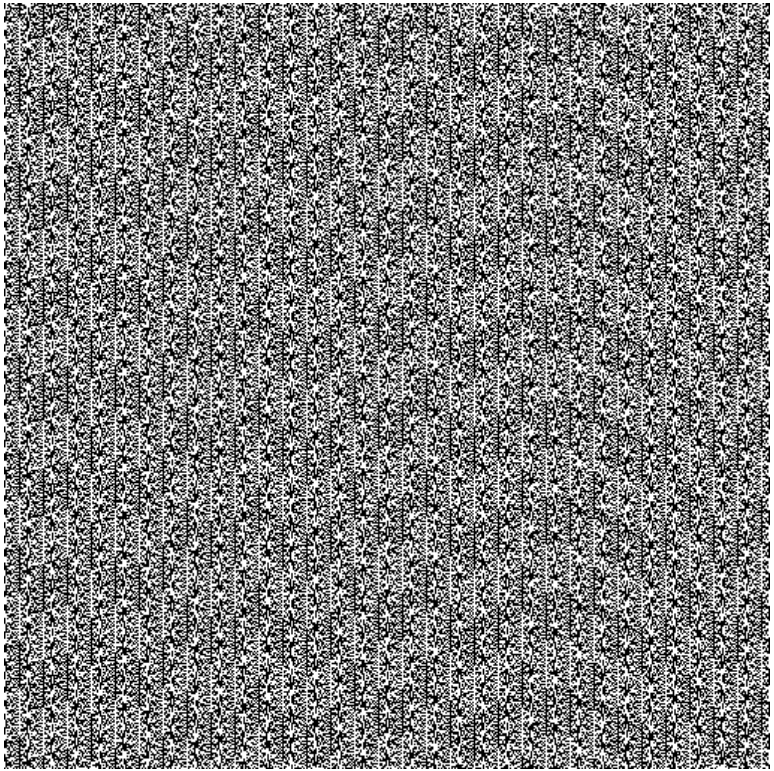
In this talk: non-secure PRNGs

- May be vulnerable to cryptographic attacks
- (Uniform) distribution guaranteed
- Predictive: same seed = same result
- Lots of seed (internal state) choices
- Long period: no intelligible patterns

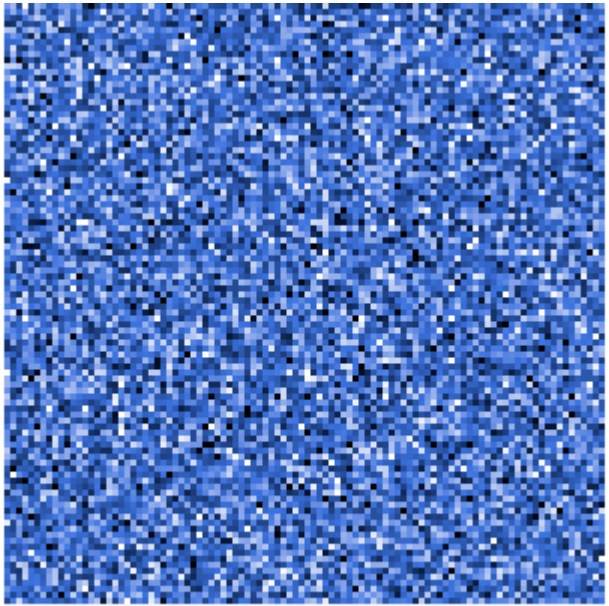
Even non-secure PRNGs fail

- Found from the observable patterns by making a graphical representation
- Very short period of showing up the same number sequence again
- Even a fairly long sequence of numbers can be fully exploited and made predictable

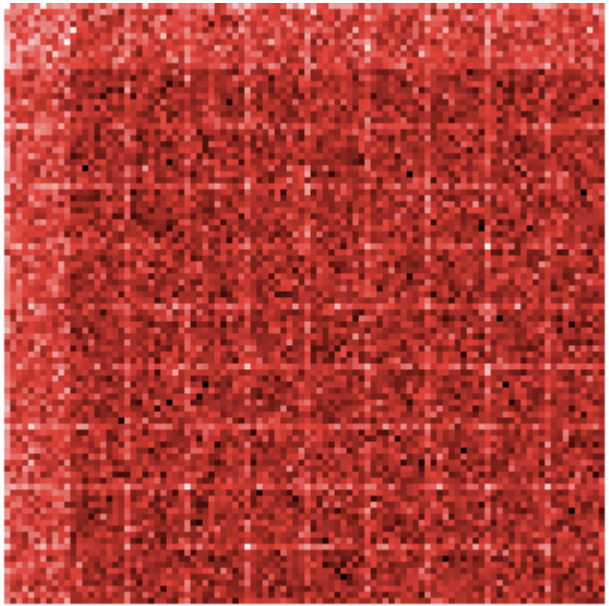
PHP rand(), as usual, PHP is an example of what not to do



Cryptocat



Colourmap of 20,000,000 Cryptocat floats (derived from /dev/urandom values 0..249)



Colourmap of 20,000,000 old-school Cryptocat floats (derived from PRNG values 0..250)

Erlang random's problem

- The algorithm AS183 is too old (designed in 1980s for 16- bit computers)
- Period: 6953607871644 ~= 2^(42.661), too short for modern computer exploits
- Fully exploited in < 9 hours on Core i5 (single core) (my C source) - Richard O'Keefe told me this was nothing new.

Alternative Erlang PRNGs

- sfmt-erlang (SFMT, 2^19937-1, 32-bit)
- tinymt-erlang (TinyMT, 2^127-1, ~2^56 orthogonal sequences, 32-bit)
- exs64 (XorShift*64, 2^64-1, 64-bit)
- exsplus (Xorshift+128, 2^128-1, 64-bit)
- exs1024 (Xorshift*1024, 2^1024-1, 64-bit)