The BEAM community and efene

Mariano Guerra @warianoguerra

efene.org @EfeneLang

instadeq.com @instadeq

Erlang User Conference

Stockholm 2017

The BEAM community and efene

Mariano Guerra @warianoguerra

efene.org @EfeneLang

instadeq.com @instadeq

Erlang User Conference

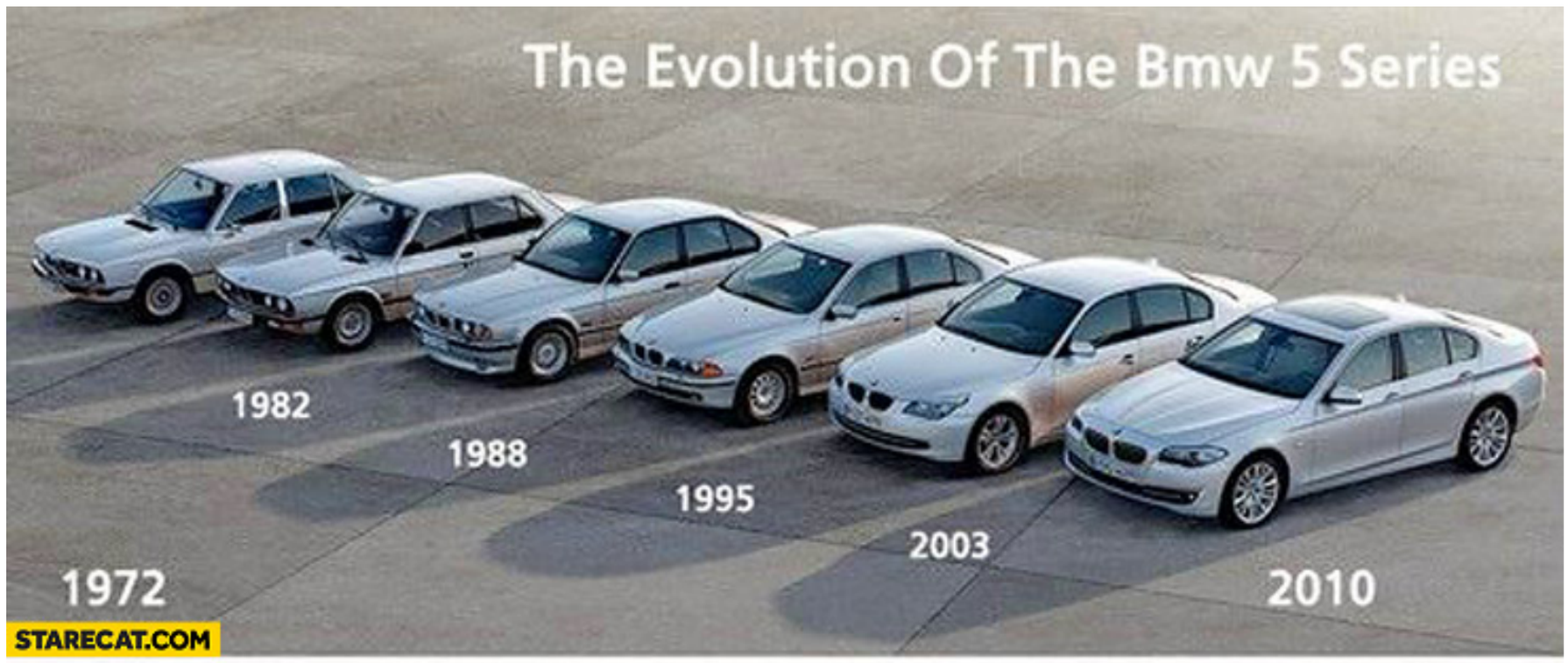
Stockholm 2017

Disclaimer

There's light at the end of the tunnel

Erlang had it 25 years ago

Fast Forward 25 years

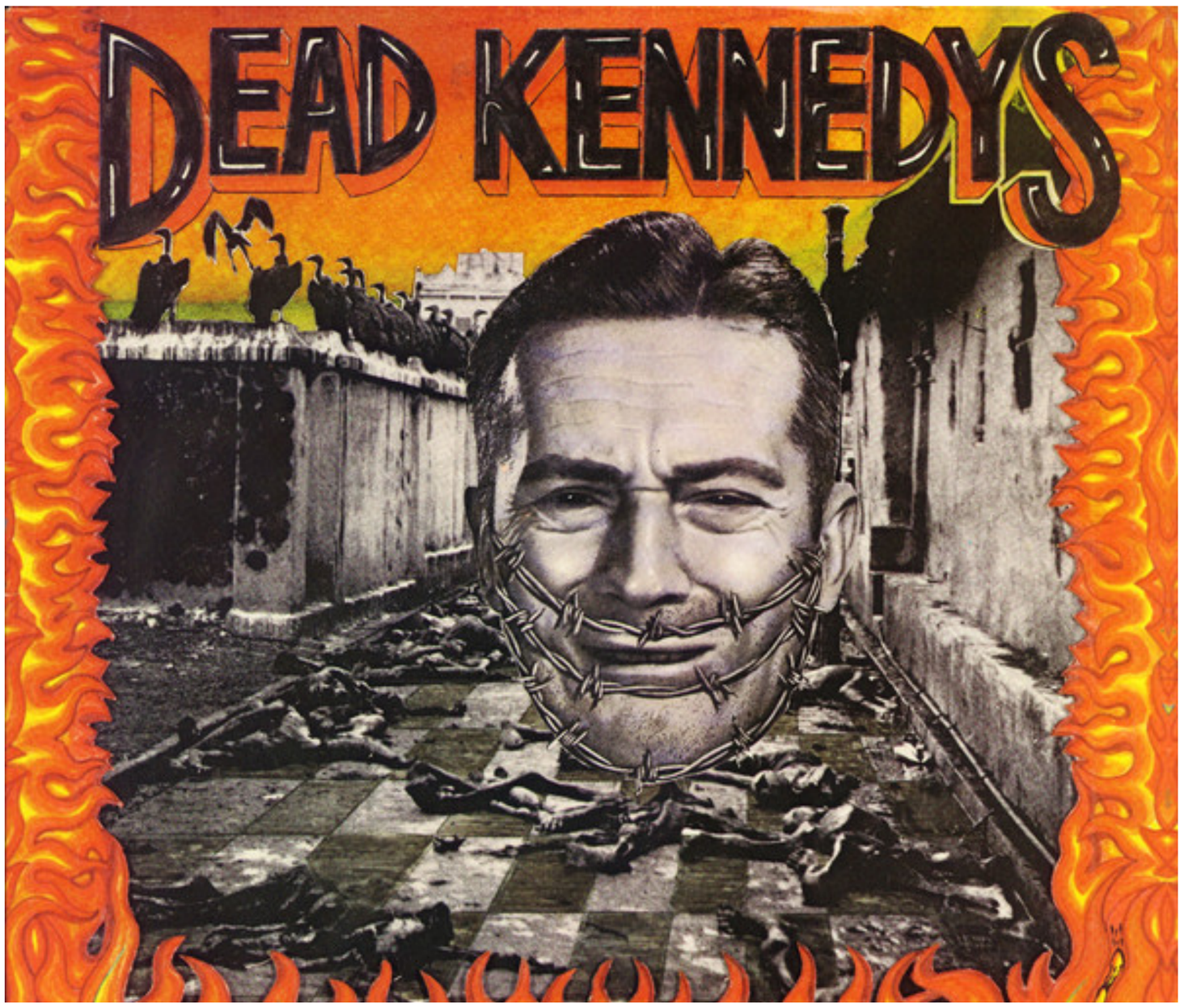


What does Erlang have in 2017 that others will copy in 2 years?

The Times They Are A-changin'


2011 Mii
1SF 0673
WOB UP 211

Give me Convenience or Give me Death




IBM

But the Smalltalk was very slow—so slow that I used to ta
coffee break while it was garbage collecting

-- Joe Armstrong

(15:11:12) DHH: before fastthread we had ~400 restarts/da

(15:11:22) DHH: now we have perhaps 10

We can tolerate many things if the tool makes us more **productive**

McCarthy wasn't using Fortran

Joe wasn't using C

DHH wasn't using **Erlang**

Productivity is not only the language semantics or the V
implementation.

It's the whole *experience*

Productivity is not only the language semantics or the V
implementation.

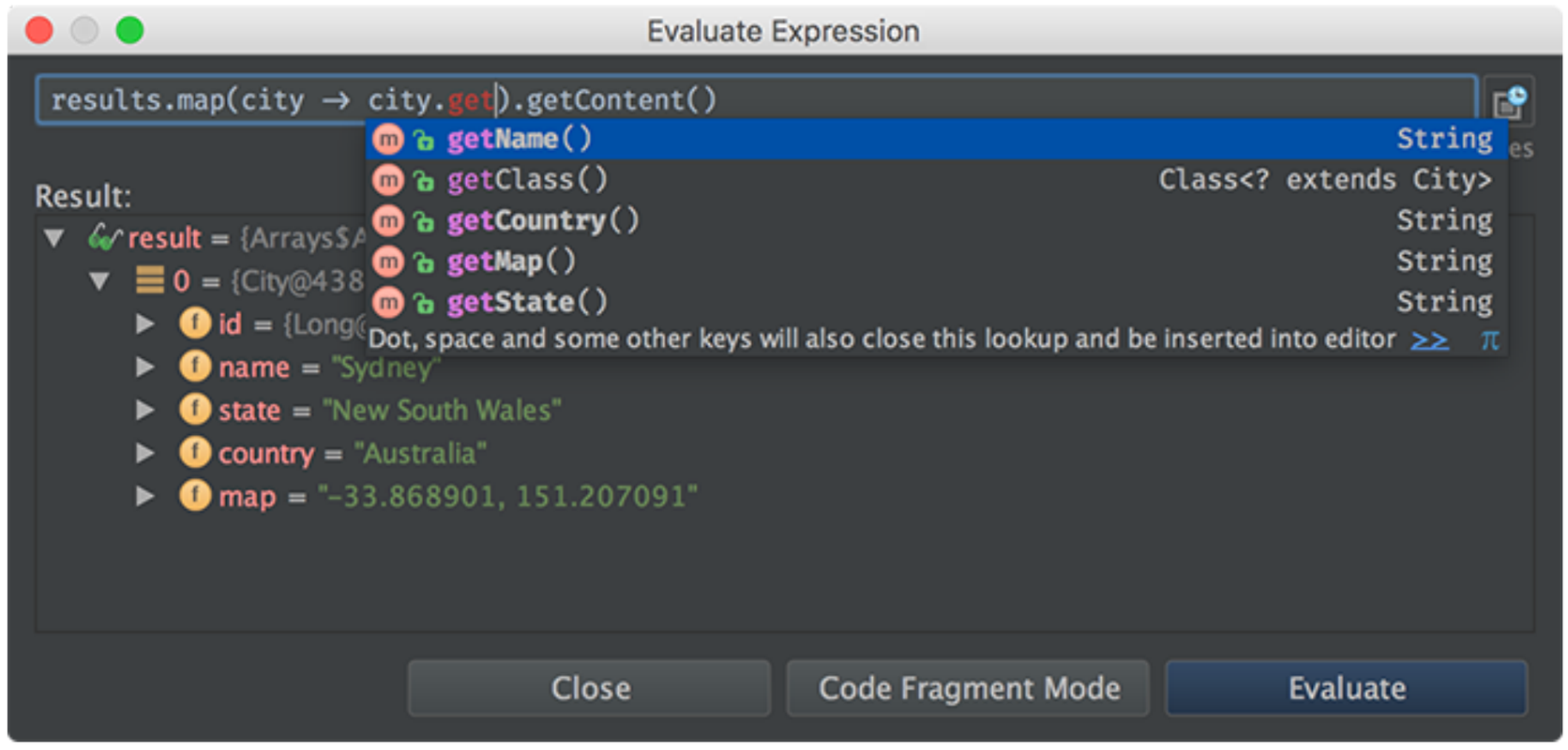
It's the whole *experience*

That's what I call **community** in this talk

The effort on **community** is inversely proportional to th
power of the tool

`-- Mariano's Law`

We call it a REPL



The Pragmatics

C -> Algol

C++/Java -> Simula/Smalltalk

Python -> ABC

Ruby -> Smalltalk

Javascript -> Self/Scheme

If we build it they will come

C -> **Algol**

C++/Java -> **Simula/Smalltalk**

Python -> **ABC**

Ruby -> **Smalltalk**

Javascript -> **Self/Scheme**

"We were after the C++ programmers. We managed to dra
lot of them about halfway to Lisp."

Guy Steele

IRC vs Slack

Web/Blogs/RSS vs Medium/Facebook

Mastodon vs Twitter

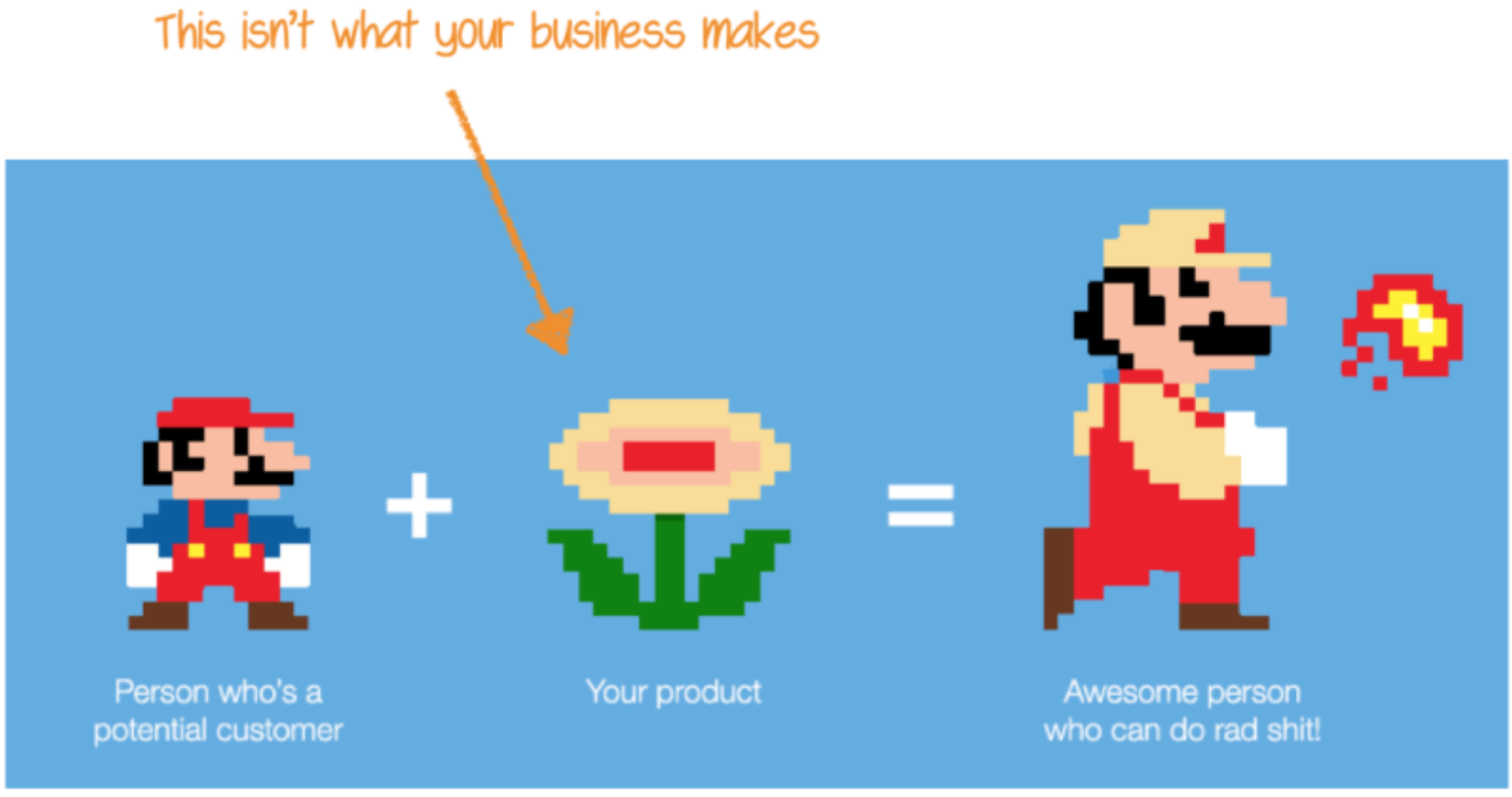
Postgres vs MySQL

Riak vs Mongo

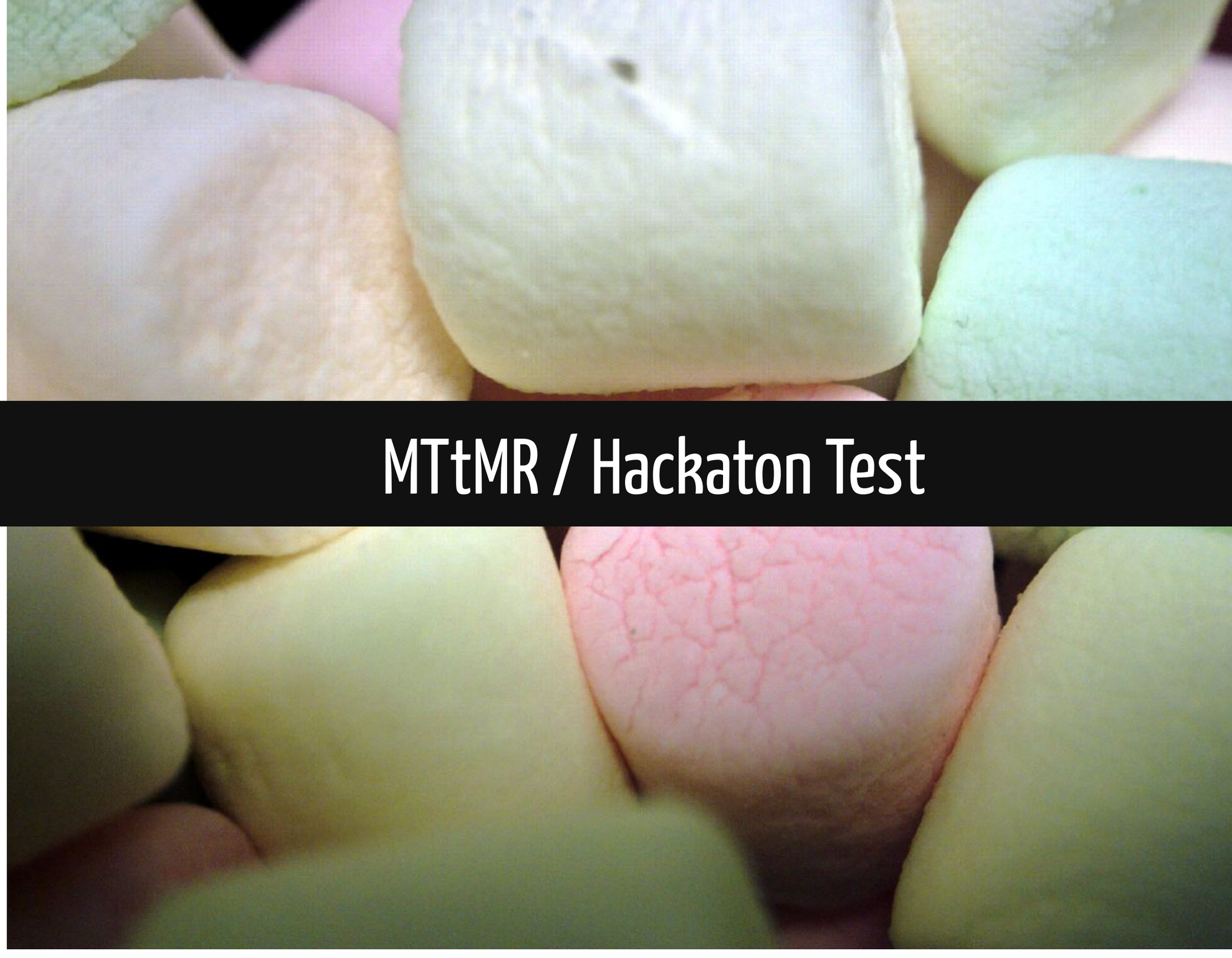
BSD vs Linux

Gmail/Google/WhatsApp etc

We are selling Erlang the wrong way



@UserOnboard



MTtMR / Hackaton Test

"Here's what our product can do" and

"Here's what you can do with our product" sound sim

but they are completely different approaches.

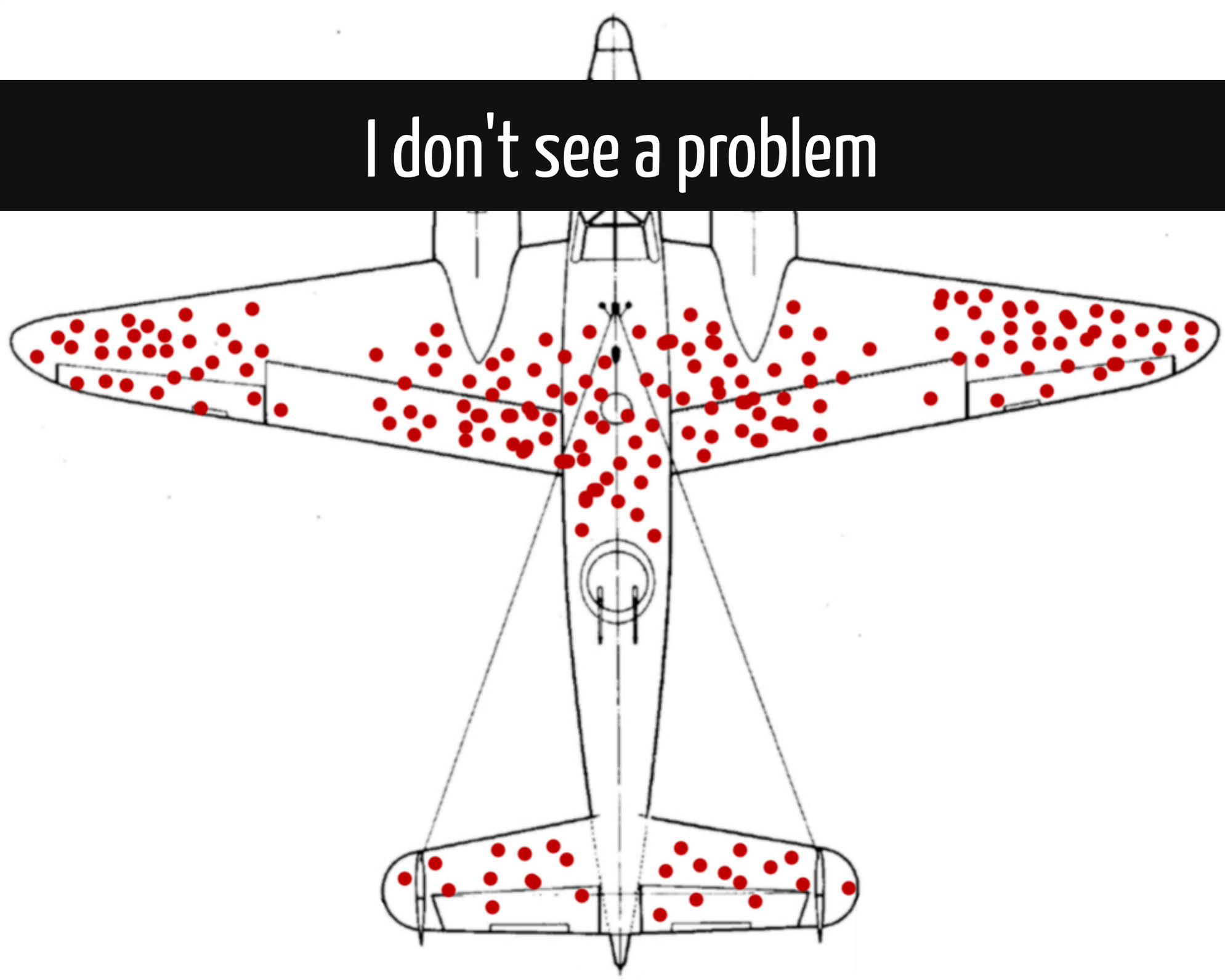
-- Jason Fried

Beware of the Powerful Language tar-pit in which everyth
is possible but nothing of interest is easy.

The Language of the System

Innovator's dilemma

I don't see a problem



Why would I care?

Bus Factor exercise:

- ferd
- t_sloughter
- lhoguin
- bytemeorg
- ostinelli
- elbrujohalcon
- heinz_gies
- gar1t
- jlouis

What if we had twice of those? 5x? 10x?

Build massively scalable soft real-time systems

Download Erlang/OTP

NEWS

Erlang/OTP 20.0-rc2 is available for testing

31 May 2017

Erlang/OTP Release Candidate 2 is available for testing

Erlang/OTP 20.0-rc1 is available for testing

5 May 2017

Erlang/OTP Release Candidate 1 is available for testing

Reporting a Security Issue in Erlang/OTP

21 Mar 2017

Use erlang-security [at] erlang [dot] org to report a security issue

GETTING STARTED

What is Erlang?

Erlang is a programming language used to build massively scal
systems with requirements on high availability. Some of its uses
banking, e-commerce, computer telephony and instant messag
system has built-in support for concurrency, distribution and fau

What is OTP?

OTP is set of Erlang libraries and design principles providing mi
develop these systems. It includes its own distributed database,
interface towards other languages, debugging and release han

Gettin

Getting Started with Erlang User's Guid

Version 8.3

March 14, 2017

Haskell

An advanced, purely functional programming language

Declarative, statically typed code.

```
primes = filterPrime [2..]
  where filterPrime (p:xs) =
          p : filterPrime [x | x <- xs, x `mod` p /= 0
```

Try it!

Type Haskell expressions in here.

λ

Got 5 minutes?

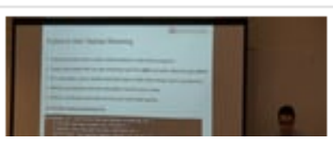
Type `help` to start the tutorial.

Or try typing these out and see what happens (click to insert):

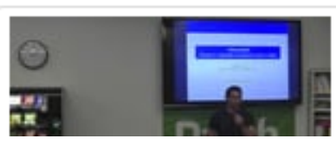
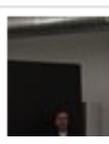
`23 * 36` or `reverse "hello"` or `foldr (:) [] [1,2,3]` or `do lin`
`putStrLn line` or `readFile "/welcome"`

These IO actions are supported in this sandbox.

Videos



Haskell's motto: Avoid success at all costs

Check Elm, OCaml, Racket sites too

Enough Complaining

- Erlang docs style improvement
    - Blogged How-To
    - Merged \o/
- Erlang client side index for search
- Erlang docs to restructured text/sphinx
    - Generic platform: from docs to X
    - beamdocs

- Applications
- Modules

- Expand All
- Contract All

- System Documentation

Application Groups

- Basic
- Database
- Operation & Maintenance
- Interface and Communication
- Tools
- Test
- Documentation
- Object Request Broker & IDL
- Miscellaneous

Erlang/OTP 20.0-rc0

Welcome to Erlang/OTP, a complete development environment for concurrent programming.

Some hints that may get you started faster

- The Erlang language is described in the Erlang Reference Manual. An Erlang tutorial can be found in Getting Started With Erlang.

  In addition to the documentation here Erlang is described in several recent books like:

  - "Introducing Erlang" from O'Reilly.
  - "Learn You Some Erlang for Great Good!" from No Starch Press.
  - "Erlang Programming" from O'Reilly.
  - "Programming Erlang" from Pragmatic.
  - "Erlang and OTP in Action" from Manning.
  - "Designing for Scalability with Erlang/OTP" from O'Reilly.

  These books are highly recommended as a start for learning Erlang.

- Erlang/OTP is divided into a number of OTP applications. An application normally contains Erlang modules. Some OTP applications, such as the C interface *erl_interface*, are written in other languages and have no Erlang modules.

- On a Unix system you can view the manual pages from the command line using

  ```
  % erl -man <module>
  ```

- You can of course use any editor you like to write Erlang programs, but if you use Emacs there exists editing support such as indentation, syntax

app

Application resource file.

The *application resource file* specifies the resources an application uses, and how the application is started. There must always be one application resource file called **Application.app** for each application **Application** in the system.

The file is read by the application controller when an application is loaded/started. It is also used by the functions in **systools** , for example when generating start scripts.

File Syntax

The application resource file is to be called **Application.app** , where **Application** is the application name. The file is to be located in directory **ebin** for the application.

The file must contain a single Erlang term, which is called an *application specification* :

```
{application, Application,
  [{description,  Description},
   {id,           Id},
   {vsn,          Vsn},
   {modules,      Modules},
   {maxP,         MaxP},
   {maxT,         MaxT},
   {registered,   Names},
   {included_applications, Apps},
   {applications, Apps},
```

and efene :)

efene is an attitude towards community, docs, tooli
UX

... and also an alternative syntax for Erlang

```
{
        "string": "sure",
        "integer": 42,
        "float": 3.14,
        "bool": true,
        "null": nil,
        "list": [1, ["nested!"]],
        "trailing ,": true,
}
```

```
{
    what: `an atom`,
    'wait!': 'binary string',
    cons: 1 :: 2 :: [],
    tuples: {
        empty: (),
        one_item: (1,),
        two: (1, 2),
    }
}
```

```
#b {Version:4, IHL:4,
    TypeOfService:8,
    TotalLength:16,
    Identification:16,
    FlagX:1, FlagD:1, FlagM:1,
    FragmentOffset:13, TTL:8,
    Protocol:8,
    HeaderCheckSum:16,
    SourceAddress:32,
    DestinationAddress:32,
    Rest: binary} = Packet
```

```
1 + -2 - 3 * 4 / 5 % 6 // 7
```

```
1 | ~2 ^ 3 & 4 >> 5 << 6
```

```
true and false or not true
```

```
1 < 2 and 2 <= 3 and 3 == 3
4 > 3 and 4 >= 4 and 4 != 3
1 is 1 and 2 isnt 3
```

```
match foo():
  case error, E1: E1
  case exit,  E2: E2
  case EType, E3 when is_tuple(E3): E3
  case EType, E4: E4
  else: (error, no_match)
end
```

```
receive
  case error, E1: E1
  case exit,  E2: E2
  case EType, E3 when is_tuple(E3): E3
  case EType, E4: E4
  else: (error, no_match)
end
```

```
fn top_level_function
  case error, E1: E1
  case exit,  E2: E2
  case EType, E3 when is_tuple(E3): E3
  case EType, E4: E4
  else: (error, no_match)
end
```

```
fn top_level_function @public
  case error, E1: E1
  case exit,  E2: E2
  case EType, E3 when is_tuple(E3): E3
  case EType, E4: E4
  else: (error, no_match)
end
```

```
fn
  case error, E1: E1
  case exit,  E2: E2
  case EType, E3 when is_tuple(E3): E3
  case EType, E4: E4
  else: (error, no_match)
end
```

```
fn NamedLambdaFun
  case error, E1: E1
  case exit,  E2: E2
  case EType, E3 when is_tuple(E3): E3
  case EType, E4: E4
  else: (error, no_match)
end
```

```
lists.map(List)  <-
  case error, E1: E1
  case exit,  E2: E2
  case EType, E3 when is_tuple(E3): E3
  case EType, E4: E4
  else: (error, no_match)
end
```

```
mlist.map(List) <<-
  case error, E1: E1
  case exit,  E2: E2
  case EType, E3 when is_tuple(E3): E3
  case EType, E4: E4
  else: (error, no_match)
end
```

```
try
  foo()
catch
  case error, E1: E1
  case exit,  E2: E2
  case EType, E3 when is_tuple(E3): E3
  case EType, E4: E4
  else: (error, no_match)
end
```

```
lists.seq(1, 10) ->>
  lists.filter(IsOdd) ->
  MyFun(param)
```

```
for X in lists.seq(1, 10):
  X + 1
end
```

```
for X in lists.seq(1, 10);
    when X % 2 is 0:
  X + 1
end
```

```
for X in lists.seq(1,  10);
    Y in lists.seq(10, 20):
  (X, Y)
end
```

```
when A < 10:
  io.format("A < 10")
else A < 20:
  io.format("A < 20 and >= 10")
else:
  io.format("A >= 20")
end
```

```
@record(acc) -> (user, pass, email)

Acc = #r.acc {user: 'bob', pass: '*'}
#r.acc {user: Uname, email: Em} = Acc
Uname = #r.acc.user Acc
Acc2 = #r.acc Acc#{email: 'bob@m.com'}
UnameIdx = #r.acc user
```

```
@type(result(E, V)) ->
    (error, E) or (ok, V)
@type(division_error()) ->
    result(division_by_zero, number())

fn divide_two @public
  @spec(number(), number()) ->
      division_error()
  case _A, 0: (error, division_by_zero)
  case A, B: (ok, A / B)
end
```

Support

- Rebar3
- Xref
- Dyalizer
- All Erlang warnings

Support

- Compile to bytecode
- Transpile 1:1 to Erlang
- Erlang hrl includes
- Erlang macros support
- No install, 2 lines in rebar.config
  - Start with one file
  - Bail out at any time
- Include fn overrides

A Boring Language

- Probably last release
  - No new syntax
- No stdlib
  - But one way of doing things
- No wrappers
- No efene ports
- No efene only tools
- Slack: a channel on Erlang slack

An Exiting Language

- Let's get things done
- Focus on docs, onboarding, templates, error msgs, community, examples, tools
- Standard stack, focus on polish and integration
  - Rebar3, Relx, Cuttlefish, Cowboy, Lager, Exometer, Re
    Syn, Shotgun, Katana, worker_pool, Common Test
- Lang Interop: Rust (cnode, NIF), Clojure (JVM), JS (Web),
  Wasm (perf)
- Sys Interop: docker/rkt/k8s, Heroku/Google/AWS/Azure,
  Windows

Status

- 1.0 Beta 1 today!
- Stable for a long time
- Been using it for tools
- Also for instadeq.com

Help!

- Try efene, read the docs
  - Open issues
  - Talk about your experience
- Syntax Highlighter
- Improve Pretty Printer
- Improve REPL
- VS Code Support
  - BEAM Language Server
- Templates
- Let's build great BEAM tooling with efene

Thanks!

Mariano Guerra @warianoguerra

efene.org @EfeneLang

instadeq.com @instadeq

What you can also do

- Can we get the cool Elixir features in the BEAM?
- Mentoring
- Blog more
- Erlang module of the week
- Clojure-docs style documentation
- Follow Erlang on Reddit, SO, HN, Twitter, #learning@Sl
and provide help

What you can also do

- Guides for common tasks
  - REST API
  - Talk to Postgres
  - Handle files
  - CLI
  - Make HTTP requests/talk to APIs
  - Binary pattern matching
- Annual survey
- Improve string module
- More