elixir

@elixirlang / elixir-lang.org

In January 2017,
Elixir became 5 years old.

2008

4 Thread Safety

The work done to make Rails thread-safe is rolling out in Rails 2.2. Dep
infrastructure, this means you can handle more requests with fewer cop
leading to better server performance and higher utilization of multiple c

To enable multithreaded dispatching in production mode of your applic
in your `config/environments/production.rb`:

```
config.threadsafe!
```

- More information :
  - Thread safety for your Rails
  - Thread safety project announcement
  - Q/A: What Thread-safe Rails Means

Functional Programming

- Explicit instead of implicit state
- Transformation instead of mutation

The Pragmatic Programmers

Seven Languages in Seven Weeks

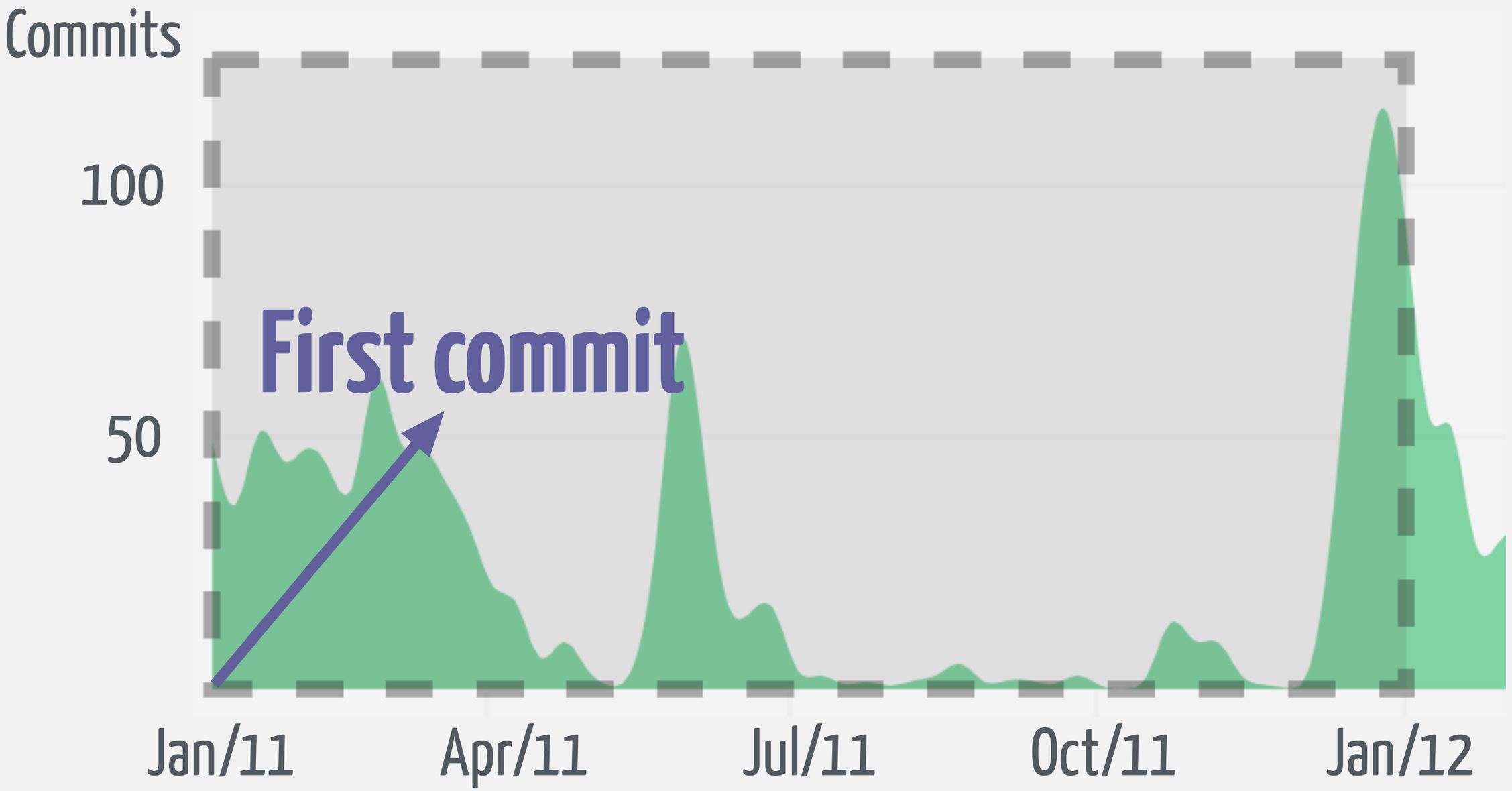
A Pragmatic Guide to Learning Programming Languages

Bruce A. Tate

Edited by Jacquelyn Carter

ERLANG

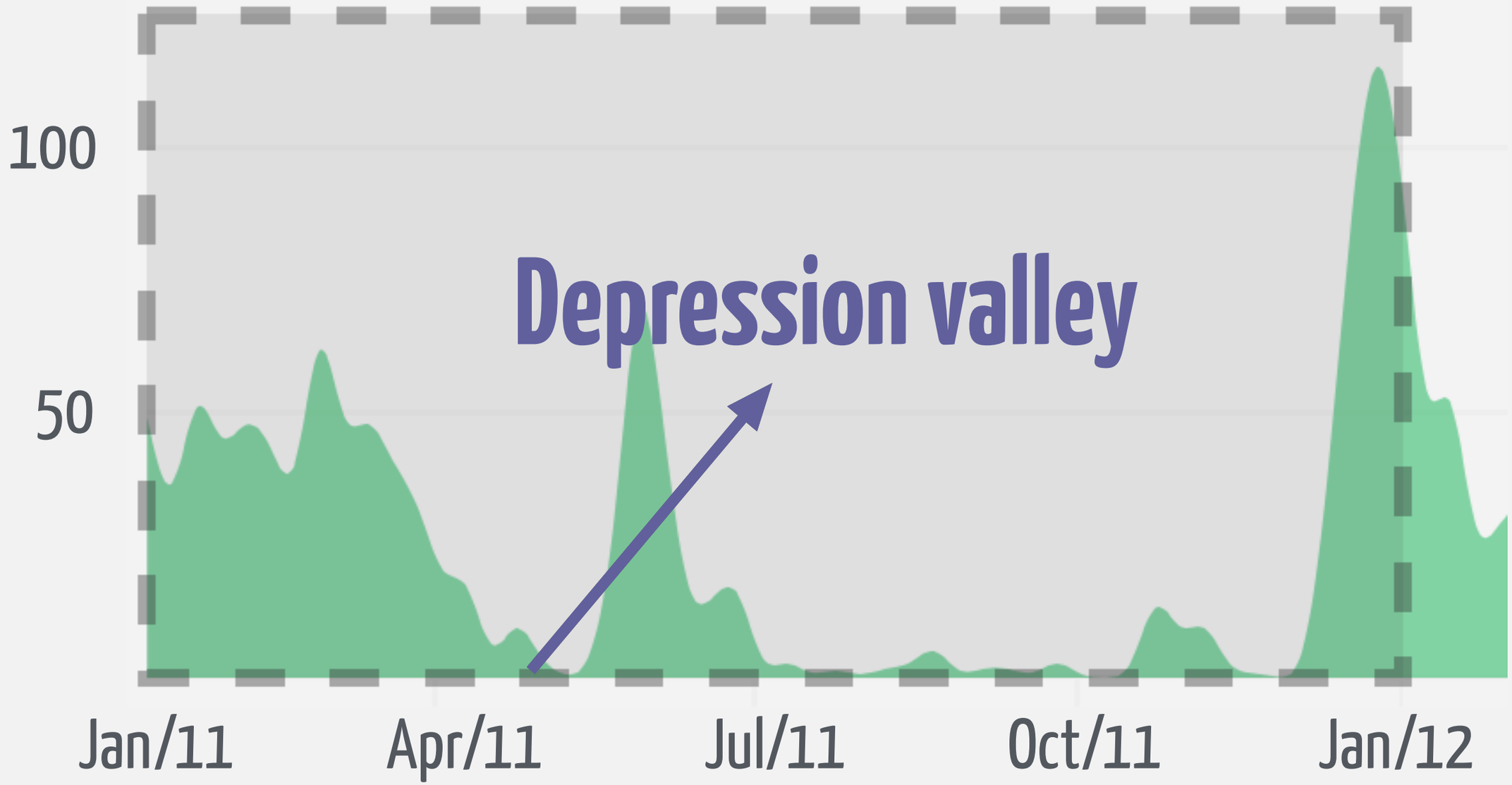
Timeline: 2011



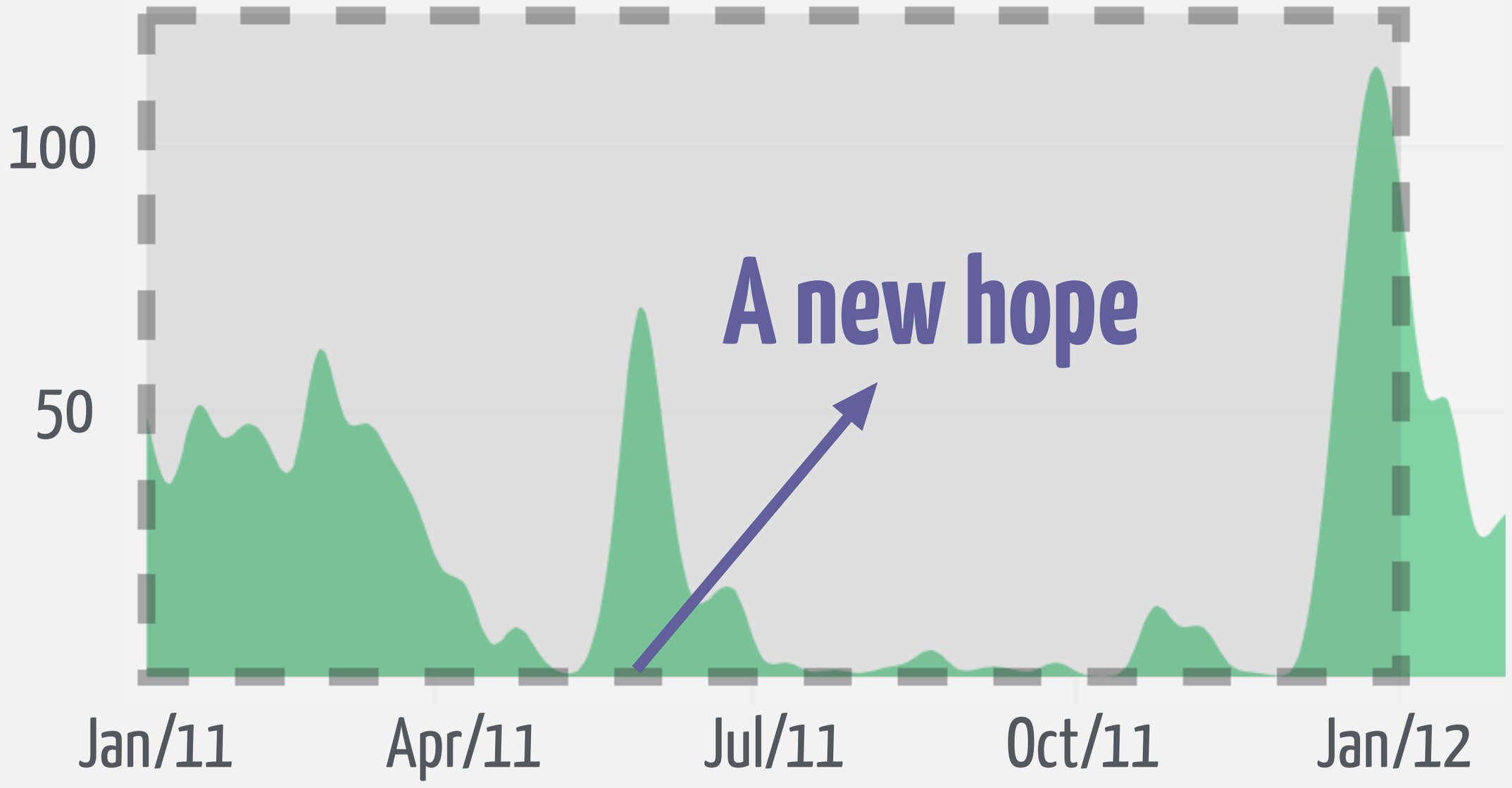
"Elixir" as of Apr/2011

- defobject to define “objects"
- prototype object-model
- eval everywhere
- slow, extremely slow
- it didn’t play well with Erlang

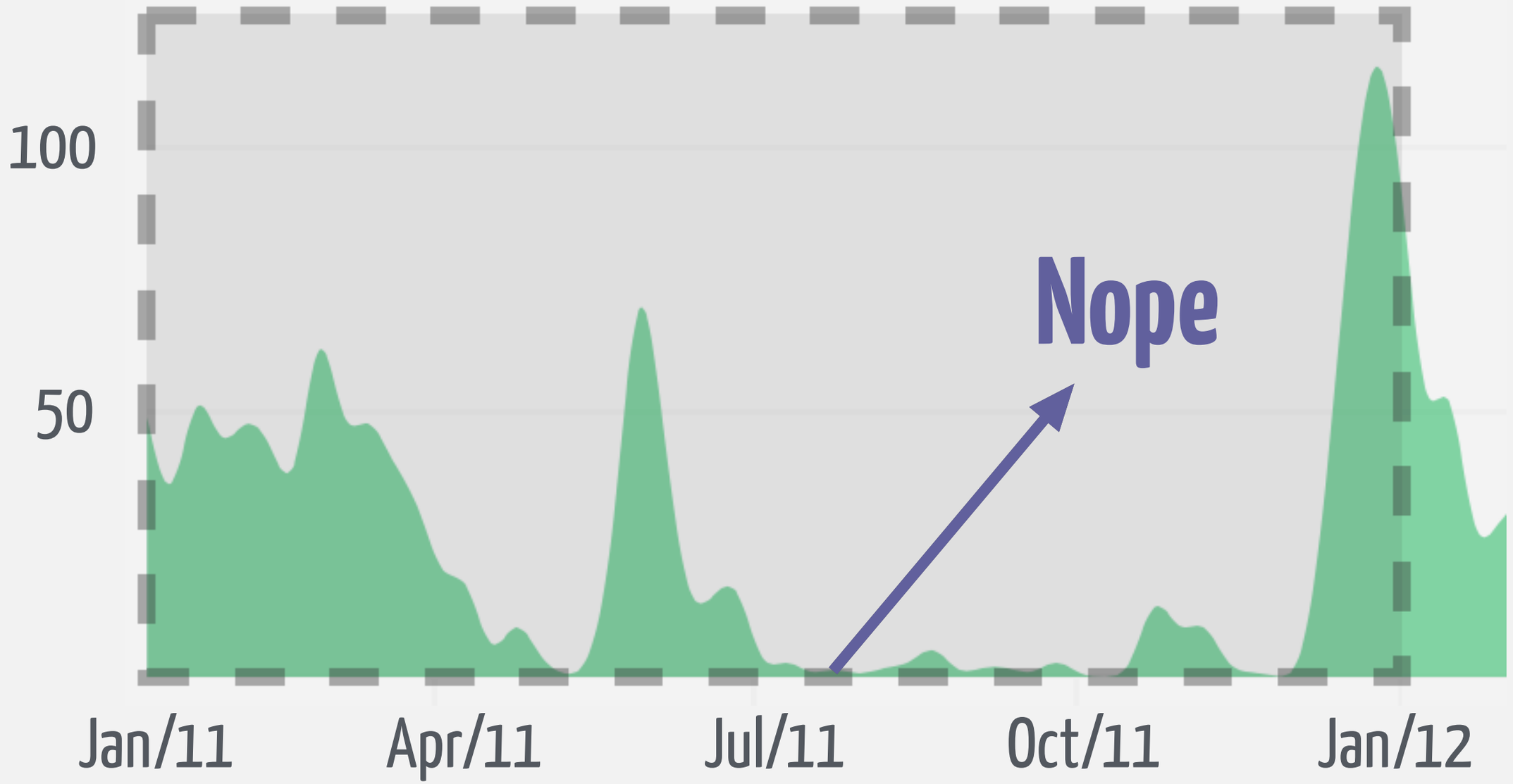
Timeline: 2011



Timeline: 2011



Timeline: 2011

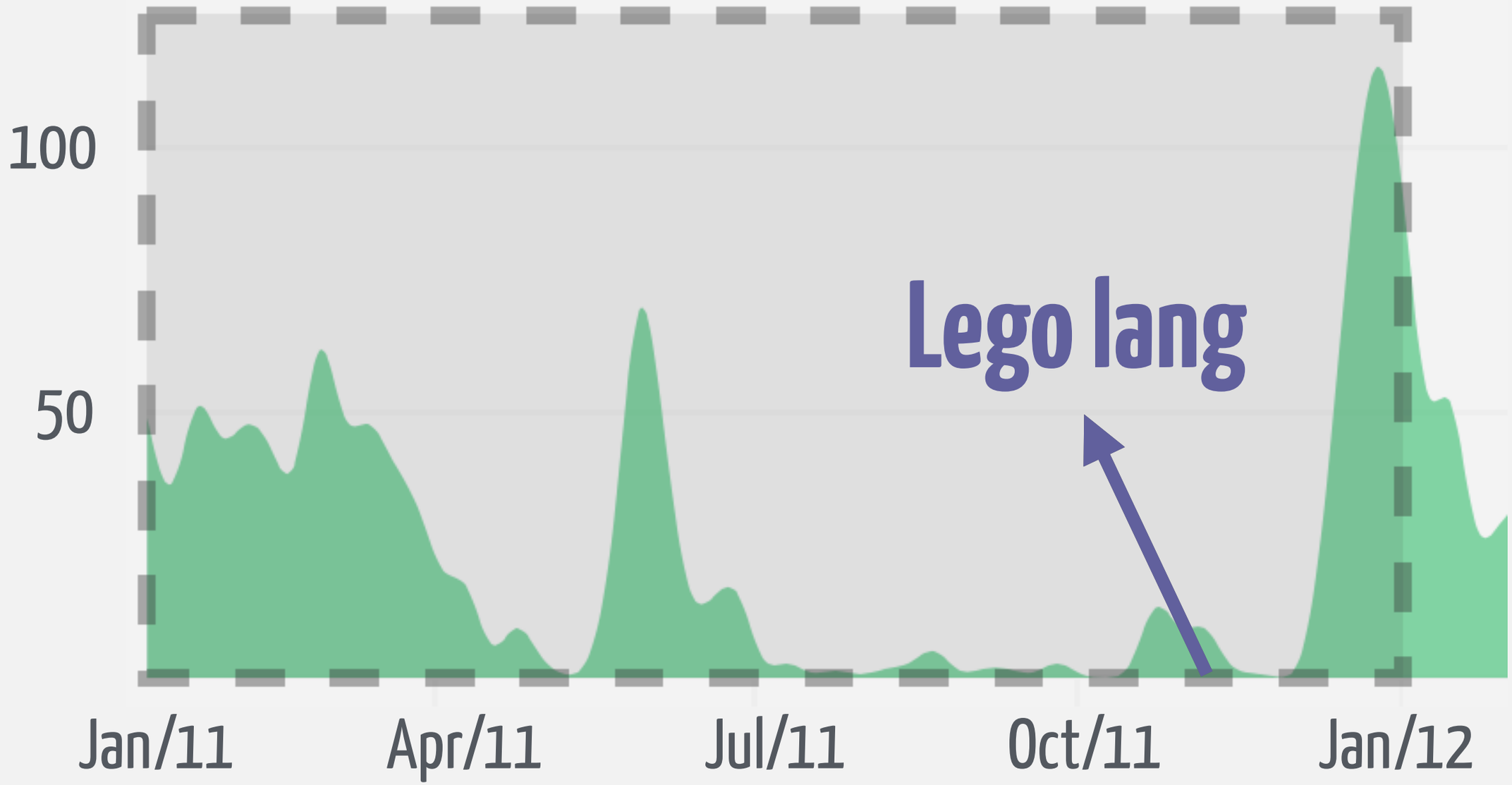



A D C B F F C
N T J K L M H
Q P O R S T
U U M A Y Z

Elixir Goals

- Productivity
  - Tooling
- Extensibility
  - Polymorphism & meta-programming
- Compatibility

Timeline: 2011



Meta-programming

- Macros are flexible
- How to combine:
  - Lisp-macros
  - Natural syntax?
- How to guarantee explicitness?

Meta-programming

```
add(1, 2)
{:add, [], [1, 2]}
```

```
add 1, 2
{:add, [], [1, 2]}
```

```
1 + 2
{:+, [], [1, 2]}
```

Meta-programming

```
defmacro unless(expr, opts) do
  quote do
    if(!unquote(expr),
       unquote(opts))
  end
end
```

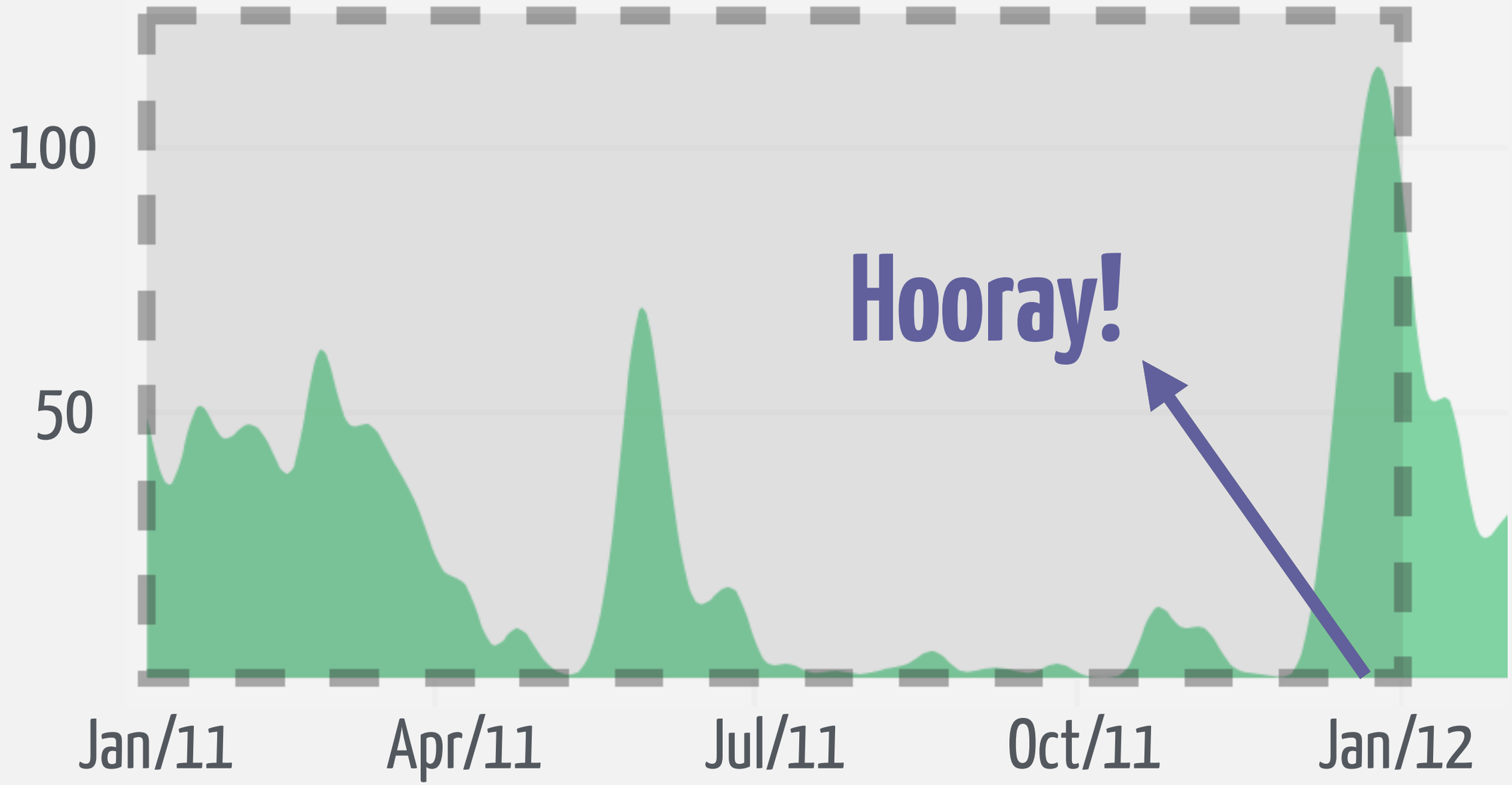
Meta-programming

```
require MyMacros
```

Macros are lexical.

Macros are compile-time only.

Timeline: 2011



plataformatec

consulting and software engineering

2012-2017

Elixir v1.0

- September/2014
- >180 contributors
- 3 books almost out
- First Elixirconf!

Elixir v1.4

- January/2017
- >520 contributors
- ~ 3800 packages on hex.pm
- Books, podcasts, online classes, etc
- Elixir events around the world!

What is Elixir?

(ElixirConf version)

Elixir is a functional, concurrent and distributed programming language for writing maintainable and scalable applications.

Data (types)

Modules

Processes

Some mutable OO language

State

Behaviour

```
dictionary.store("key", “value")
```

Time

Time

Time

State

Behaviour

Data (types)

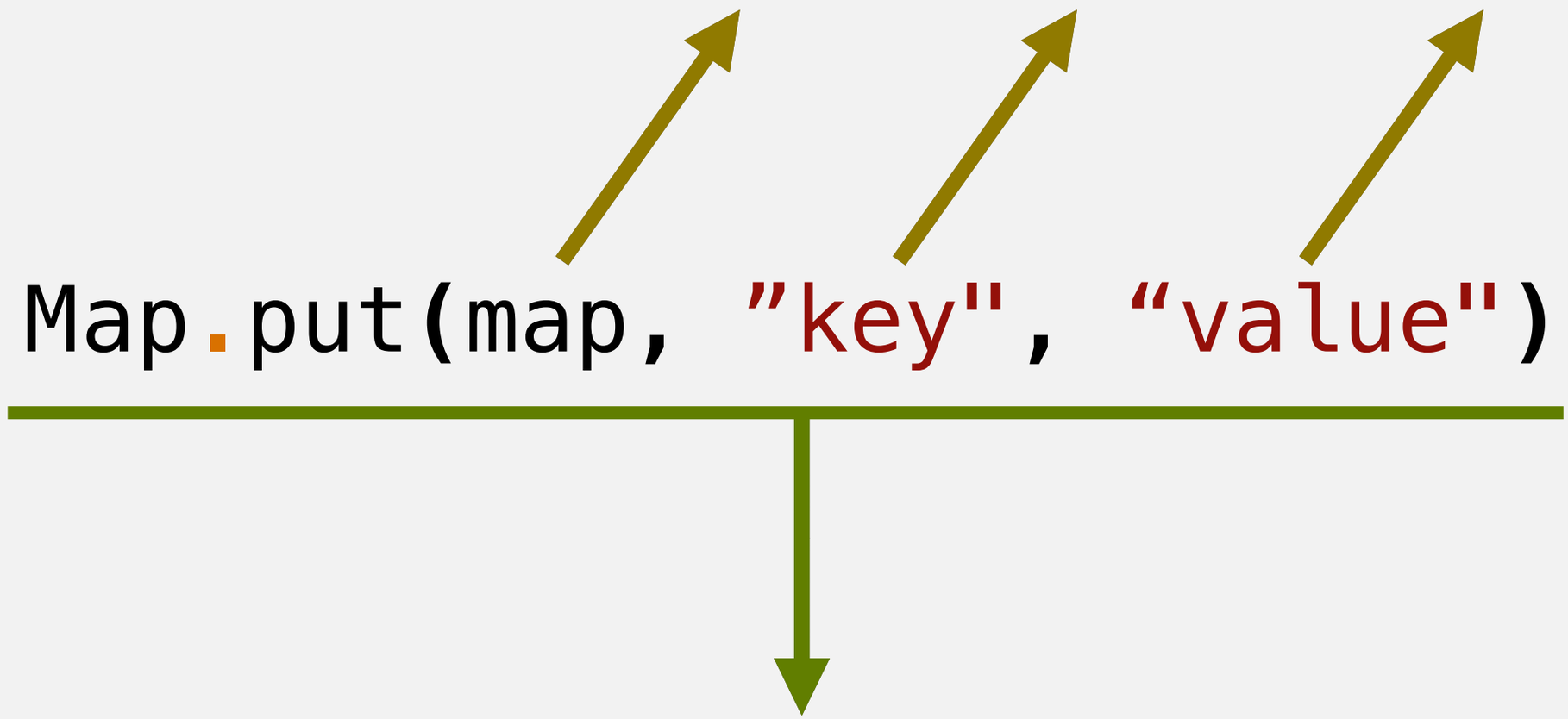
Modules

Processes

Time

Data + Modules

Data Data Data



Processes

```
spawn(Server, :loop, [initial_data])
```

Processes

Data belongs to a process

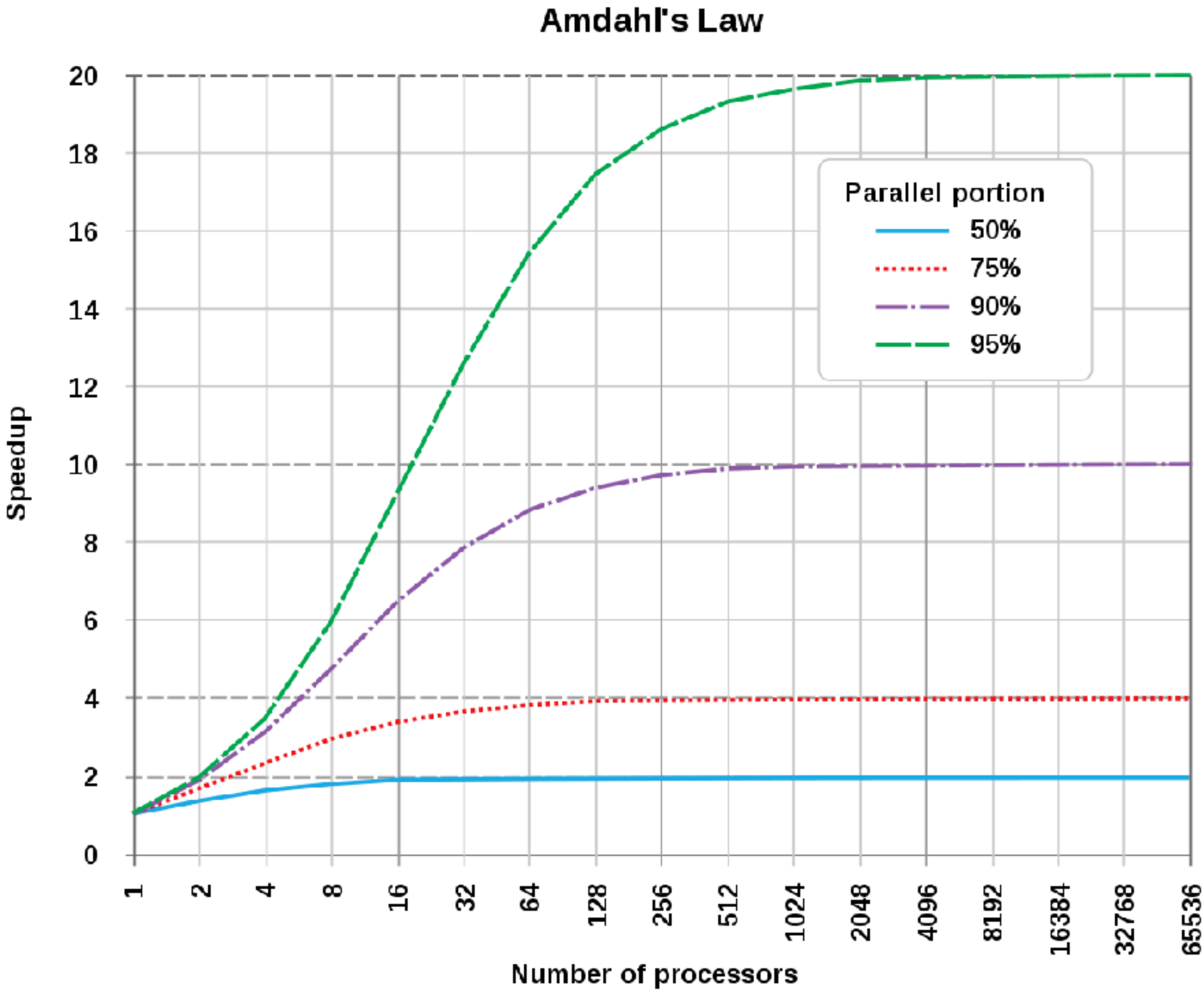
```
def loop(data) do
  # ...
end
```

Code runs sequentially

Processes

- Processes own the data
- Run code (behaviour) sequentially
- Introduce concurrency
- What about time?
  - Model time with message-passing

Concurrency without a good mental model will be constrained by Amdahl's Law


Amdahl's Law
Speedup
Number of processors
Parallel portion
50%
75%
90%
95%
0
2
4
6
8
10
12
14
16
18
20
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536

But processes go way beyond concurrency...

When to create processes?

- To manage mutable state
- For concurrent execution
- For handling and isolating failures
- For distributed communication

State

Behaviour

Data (types)

Modules

Processes

Time

Elixir v1.5

Short-term features are in the issues tracker.

Join the discussion and contribute!

Active Research Projects

GenStage & Flow

Announcing GenStage

July 14, 2016 · by José Valim . in Announcements

Today we are glad to announce the official release of GenStage. GenStage is a new Elixir behaviour for exchanging events with back-pressure between Elixir processes. In the short-term, we expect GenStage to replace the use cases for GenEvent as well as providing a composable abstraction for consuming data from third-party systems.

In this blog post we will cover the background that led us to GenStage, some example use cases, and what we are exploring for future releases. If instead you are looking for a quick reference, check the project source code and access its documentation.

Background

One of the original motivations for creating and designing Elixir was to introduce better abstractions for working with collections. Not only that, we want to provide developers interested in manipulating collections with a path to take their code from eager to lazy, to concurrent and then distributed.

Let's discuss a simple but actual example: word counting. The idea of word counting is to receive one file and count how many times each word appears in the document. Using the `Enum` module it could

http://bit.ly/genstage

UTF-8 Atoms

```
test "こんにちは世界" do
  assert :こんにちは世界
end
```

erlang / otp

Watch 516 Unstar 5,457 Fork 1,531

Code Pull requests 28 Projects 0 Wiki Pulse Graphs

Add new AtU8 beam chunk #1078

Edit

Merged bjorng merged 1 commit into `erlang:master` from `josevalim:jv-atu8-chunk` on 1 Feb

Conversation 72 Commits 1 Files changed 21 +387 −200

josevalim commented on 31 May 2016 • edited Contributor

The new chunk stores atoms encoded in UTF-8.

This is still work in progress and the following tasks are still pending:

- [x] Add the compile option `r19` that will compile atoms to the old "Atom" chunk as mentioned by @bjorng in the mailing list
- [x] Support 255 UTF-8 codepoints in `binary_to_atom` instead of 255 bytes. The issue is that calculating the number of characters is linear to the binary size. Although `erts_atom_put` performs such calculation, it returns a NON-VALUE for both invalid encoding and large binaries errors, and the test suite expects badarg for invalid encoding and system limit for large binaries. To solve this, we can either:

  i. add a new function, used by both erts_atom_put and binary_to_atom, that properly encodes

Reviewers

psyeugenic

bjorng

Assignees

bjorng

Labels

feature

kanban

team:MW

team:VM

https://github.com/erlang/otp/pull/1078

Data streams & Property testing

```elixir
test "starts_with?/2" do
  for s1 <- Data.string(),
      s2 <- Data.string() do
    String.starts_with?(s1 <> s2, s1)
  end
end
```

GenHTTP

Early Research

GenHTTP

- Model HTTP at the process-level
- Think of a wrapper around :gen_tcp
- Better foundation, more flexible

elixirfmt

Language Server Protocol

langserver.org

Built and designed at

plataformatec

consulting and software engineering

plataformatec

consulting and software engineering

Elixir coaching

Elixir design review

Custom development

elixir

@elixirlang / elixir-lang.org