Code Beam SF 2018

Elixir Core Update

Elixir

Elixir

http://elixir-lang.org

Repositories 11 | People 15 | Teams 5 | Projects 0

Pinned repositories

elixir
Elixir is a dynamic, functional language designed for building scalable and maintainable applications
Elixir ★ 12.5k 1.8k

gen_stage
Producer and consumer pipelines with back-pressure for Elixir
Elixir ★ 604 105

ex_doc
ExDoc produces HTML and EPUB documentation for Elixir projects
Elixir ★ 544 120

elixir-lang.github.com
Website for Elixir
CSS ★ 187 588

gettext
Internationalization and localization support for Elixir.
Elixir ★ 166 34

flow
Computational parallel flows on top of GenStage
Elixir ★ 410 31

Elixir

2011: Initial commit

2012: José Valim presents at Erlang Factory Krakow

2013: 1.6k GitHub stars / my first commit to Elixir

2014: v1.0 / 3.4k GitHub stars / first Elixir conference

Elixir

2015: v1.1 / 5.9k Github stars

2016: v1.2 / v1.3 / 8.9k Github stars

2017: v1.4 / v1.5 / 11.9k Github stars

2018: v1.6 / v1.7 / 12.5k+ Github stars / #CodeBEAMSF

Elixir

Semantic Versioning

Minor releases every 6 months

Backwards compatible features

Bug fixes may be breaking changes :(

Elixir

Productivity

Maintainability

Reliability

v1.6

Format

Code Formatter

Consistent style for Elixir code base

Easier to contribute

Provide immediate feedback loop

My code looks like your code! Your code looks like my code!

Code Formatter

Code.format_file!/1,2

Code.format_string!/1,2

mix format file1.ex file2.exs

Code Formatter

Before:

```
defmacro defmodule(alias, do_block)
defmacro defmodule(alias, do: block) do
  env = __CALLER__
  boot? = bootstrapped?(Macro)
  expanded = case boot? do
    true  -> Macro.expand(alias, env)
    false -> alias
  end
```

Code Formatter

After:

```
defmacro defmodule(alias, do_block)

defmacro defmodule(alias, do: block) do
  env = __CALLER__
  boot? = bootstrapped?(Macro)

  expanded =
    case boot? do
      true -> Macro.expand(alias, env)
      false -> alias
    end
```

Code Formatter

No configuration…except when you really need to

No discussion…except for any bugs (please try it and report!)

No special cases 

No changing AST 

AST - Abstract Syntax Tree

Term structure of the code

```
defmodule CodeBEAMSF do
  def hello(name) do
    quote do
      "Hello, #{name}!"
    end
  end
end
```

AST - Abstract Syntax Tree

Term structure of the code

```
{:def, [context: CodeBEAMSF, import: Kernel],
 [
   {:hello, [context: CodeBEAMSF], [{:name, [], CodeBEAMSF}]},
   [
     do: {:quote, [],
      [
        [
          do: {:<<>>, [],
           [
             "Hello, ",
             {:::, [],
              [
                {{:., [], [Kernel, :to_string]}, [], [{:name, [], CodeBEAMSF}]},
                {:binary, [], CodeBEAMSF}
              ]},
             "!"
           ]}
        ]
      ]}
   ]
 ]}
```

AST - Abstract Syntax Tree

Code.string_to_quoted/1,2

Tokenization: convert string to tokens

Parse: convert tokens to quoted (AST)

AST - Abstract Syntax Tree

Code.string_to_quoted/1,2

```
@spec string_to_quoted(List.Chars.t(), keyword) ::
        {:ok, Macro.t()} | {:error, {line :: pos_integer, term, term}}
def string_to_quoted(string, opts \\ []) when is_list(opts) do
  file = Keyword.get(opts, :file, "nofile")
  line = Keyword.get(opts, :line, 1)

  with {:ok, tokens} <- :elixir.string_to_tokens(to_charlist(string), line, file, opts) do
    :elixir.tokens_to_quoted(tokens, file, opts)
  end
end
```

v1.6

Compile

Macros

Macros are compile time dependencies

If the macro's module changes we need to recompile

require/1,2 informs the compiler and YOU!

```
Compiling 100 files (.ex)
Generated code_beam_sf app
```

mix xref

mix xref graph (v1.3)

```
$ mix xref graph
lib/ecto/repo.ex
├── lib/ecto/adapter.ex
│   ├── lib/ecto/query.ex
│   │   ├── lib/ecto/exceptions.ex
│   │   │   ├── lib/ecto/changeset.ex (struct)
│   │   │   │   ├── lib/ecto/association.ex (struct)
│   │   │   │   │   ├── lib/ecto.ex
│   │   │   │   │   │   ├── lib/ecto/association.ex (struct)
│   │   │   │   │   │   ├── lib/ecto/exceptions.ex
│   │   │   │   │   │   └── lib/ecto/schema.ex
│   │   │   │   │   │       ├── lib/ecto/association.ex (struct)
│   │   │   │   │   │       ├── lib/ecto/embedded.ex
│   │   │   │   │   │       │   ├── lib/ecto.ex
│   │   │   │   │   │       │   ├── lib/ecto/changeset.ex (struct)
│   │   │   │   │   │       │   ├── lib/ecto/changeset/relation.ex (compile)
```

mix xref

mix xref graph —format stats

```
$ mix xref graph format --stats
Tracked files: 67 (nodes)
Compile dependencies: 40 (edges)
Structs dependencies: 31 (edges)
Runtime dependencies: 170 (edges)

Top 10 files with most outgoing dependencies:
  * lib/ecto/query.ex (15)
  * lib/ecto/association.ex (14)
  * lib/ecto/repo/queryable.ex (13)
  * lib/ecto/repo/schema.ex (12)
  * lib/ecto/changeset.ex (11)
  * lib/ecto/adapters/sql.ex (8)
  * lib/ecto/adapters/postgres.ex (8)
  * lib/ecto/adapters/mysql.ex (8)
  * lib/ecto/repo/preloader.ex (7)
  * lib/ecto/repo.ex (7)
```

mix xref

```
# To get all files that depend on lib/code_beam_sf.ex
mix xref graph --sink lib/code_beam_sf.ex --only-nodes

# To get all files that depend on lib/code_beam_sf.ex at compile time
mix xref graph --label compile --sink lib/code_beam_sf.ex --only-nodes

# To get all files lib/code_beam_sf.ex depends on
mix xref graph --source lib/code_beam_sf.ex --only-nodes

# To limit statistics only to compile time dependencies
mix xref graph --format stats --label compile
```

@deprecated and @since

```elixir
defmodule CodeBEAMSF do
  @since "2.0.18"
  def hello(name), do: "Hello, #{name}!"
end

defmodule ErlangAndElixirFactorySF do
  @since "2.0.17"
  @deprecated "2.0.18"
  def hello(name), do: "Hello, #{name}!"
end

defmodule ErlangFactorySF do
  @deprecated "2.0.17"
  def hello(name), do: "Hello, #{name}!"
end
```

defguard

```elixir
defguard is_drinking_age(age) when age >= 21

def serve_drinks(%User{age: age}) when is_drinking_age(age) do
  # Code that serves drinks!
end
```

v1.6

Test

mix test —slowest <N>

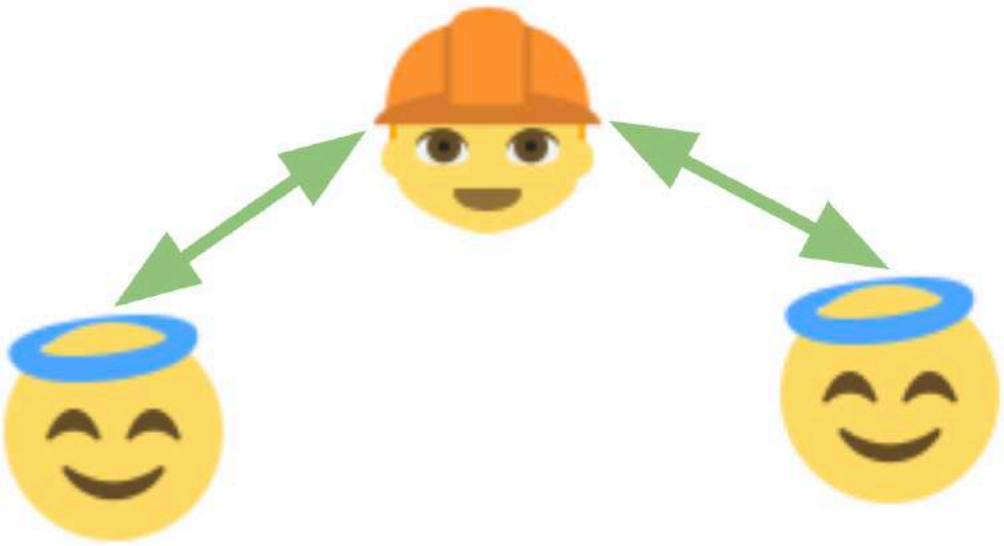
```
$ mix test --slowest 5
...
Top 5 slowest (0.4s), 18.6% of total time:

  * test http2 (120.1ms) (Plug.Adapters.Cowboy2.ConnTest)
  * test fails on large headers (81.0ms) (Plug.Adapters.Cowboy2.ConnTest)
  * test call/2 is overridden and handles errors without sources (79.0ms) (Plug.DebuggerTest)
  * test builds a connection (79.0ms) (Plug.Adapters.Cowboy2.ConnTest)
  * test call/2 is overridden and warns on non-500 errors (69.4ms) (Plug.DebuggerTest)
```

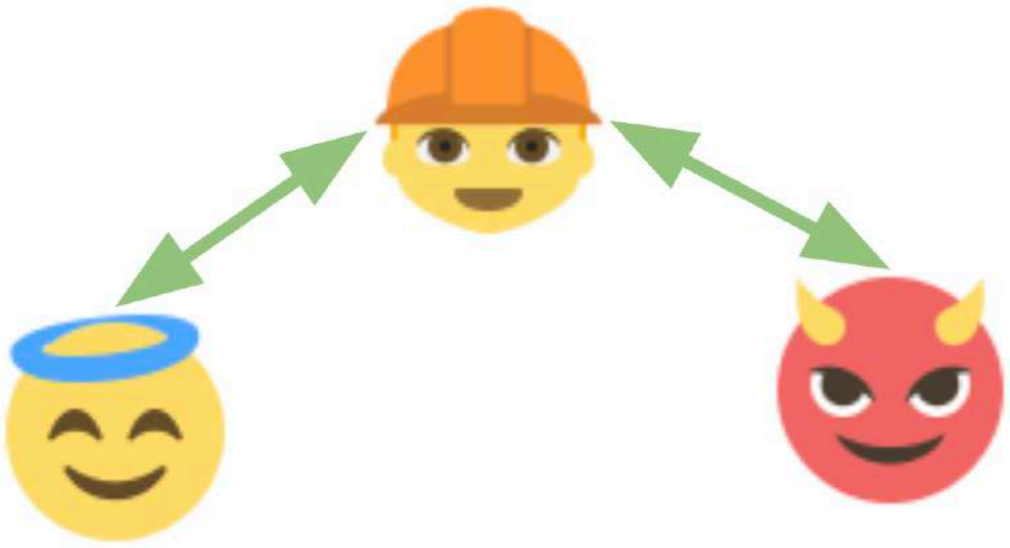
v1.6

Supervise

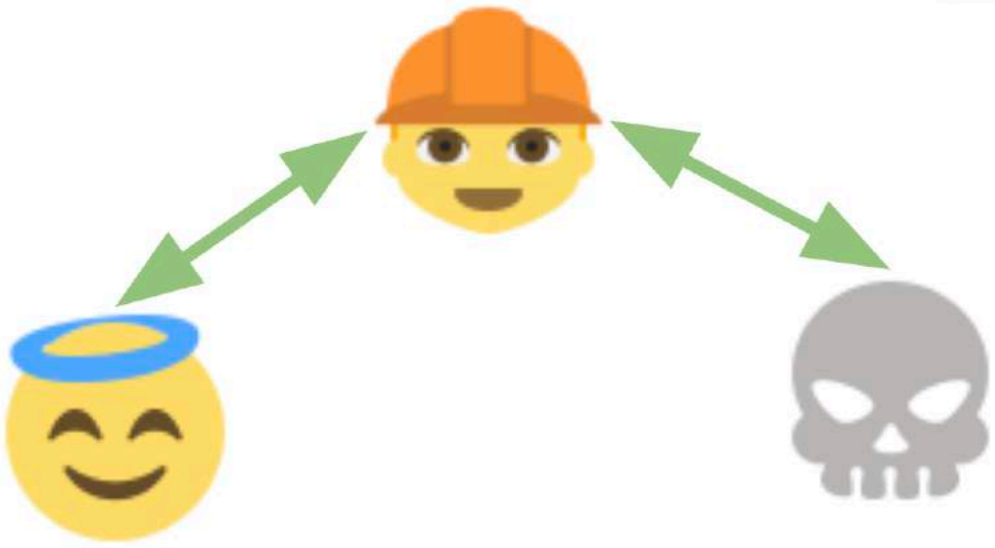
Supervisors



Supervisors



Supervisors



Supervisors

one_for_one

one_for_all

rest_for_one

simple_one_for_one

Children

child_spec/1 (v1.5)

```elixir
def child_spec(opts) do
  %{id: __MODULE__,
    start: {__MODULE__, :start_link, [opts]}}
end

Supervisor.start_link({MyModule, opts}, [strategy: :one_for_one])
```

Children

simple_one_for_one doesn't work so well

```
spec = Supervisor.child_spec({MyModule, :ignore}, start: {MyModule, :start_link, []})
{:ok, sup} = Supervisor.start_link([spec], strategy: :simple_one_for_one)
{:ok, pid} = Supervisor.start_child(sup, [opts])
```

DynamicSupervisor

Just like the simple_one_for_one but actually simple

```
def child_spec(opts) do
  %{id: __MODULE__,
    start: {__MODULE__, :start_link, [opts]}}
end

{:ok, sup} = DynamicSupervisor.start_link([strategy: :one_for_one])
DynamicSupervisor.start_child(sup, {MyModule, opts})
```

v1.6

More!

More!

Unicode 10 Support

Enum.slice/2,3

Time.add/2,3

Compiler diagnostics

v1.7

Coming soon!

Releases

Mix will generate releases

mix run and *iex -S mix* will run the release

Integrated configuration with *config.exs* files

StreamData

Data generation

```
require ExUnitProperties

domains = ["gmail.com", "hotmail.com", "yahoo.com"]

email_generator =
  ExUnitProperties.gen all name <- StreamData.string(:alphanumeric),
                           name != "",
                           domain <- StreamData.member_of(domains) do
    name <> "@" <> domain
  end

Enum.take(StreamData.resize(email_generator, 20), 2)
#=> ["efsT6Px@hotmail.com", "swEowmk7mW0VmkJDF@yahoo.com"]
```

StreamData

Property based testing

```elixir
use ExUnitProperties

property "bin1 <> bin2 always starts with bin1" do
  check all bin1 <- binary(),
            bin2 <- binary() do
    assert String.starts_with?(bin1 <> bin2, bin1)
  end
end
```

v1.8?

Research

Dialyzer support

Warnings are in Erlang format

Warnings are explicit but high learning curve

PLT management is non-trivial

Dialyzer support

Warnings in Elixir format

Additional information in warnings

Mix manages PLT

Better compilation errors

Some errors are non-trivial to understand or debug:

```
== Compilation error in file lib/code_beam_sf.ex ==
** (SyntaxError) lib/code_beam_sf.ex:620: unexpected token: end
```

Take inspiration from Elm and Rust

Better docs for all!

EEP 48 - Documentation storage and format

Standard API for all BEAM languages

Access in shell (already supported for Elixir modules)

```
iex(1)> h CodeBEAMSF.hello/1

                                  def hello(name)

Say hello!
```

Contribute

Welcome Michal

Congratulations to our new core team member!

New contributors

First time code contributions per week

Contribute

It's not about commit count to Elixir core or Erlang/OTP

Blogs and Books

Meetups and Conferences

Tutorials and Teaching

Podcasts and Evangelizing


Thank you!