Inch

HOW ELIXIR 1.7 CHANGED THE RULES FOR DOCUMENTATION ANALYSIS

René Föhring, Berlin, 2018

@rrrene

5Minds IT-SOLUTIONS

Hi, my name is René.

I work at 5Minds IT-SOLUTIONS

I'm @rrrene on Twitter/GitHub.

Credo

ame is René.

OLUTIONS

witter/GitHub.

Credo

Inch

5Minds IT-SOLUTIONS

1. Docs?
2. Tools?
3. EEP 48!

Project website

Reference material

Inline Docs

READMEs

How-to guides

Tutorials

Ahem … Inline Docs?

```
#
# TODO: write some docs
#
def size(filename_or_blob, mode \\ nil)
```

Docs in Elixir

```elixir
@doc """
TODO: write some docs
"""
def size(filename_or_blob, mode \\ nil)
```

Docs in Elixir
= First Class Citizen

```
@doc """
TODO: write some docs
"""
def size(filename_or_blob, mode \\ nil)
```

```elixir
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)
```

Markdown

```
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)
```

```elixir
@doc """
Public: Detects the size of the blob.

filename_or_blob – filename or blob
mode – Optional mode (defaults to nil)

Examples

    iex> MyModule.size(filename)
    4096

Returns an integer or `nil`.
"""
def size(filename_or_blob, mode \\ nil)
```

you could also
use Tomdoc

@rrrene

5Minds IT-SOLUTIONS

but you won‘t

```
@doc false
def size(filename_or_blob, mode \\ nil)
```

same goes for macros

```
@doc false
defmacro size(filename_or_blob, mode \\ nil)
```

… and modules

```
defmodule MyModule do
  @moduledoc false
end
```

summary

```
defmodule MyModule do
  @moduledoc "..."

  @doc "..."
  defmacro __using__(opts \\ [])

  @doc "..."
  def size(filename_or_blob, mode \\ nil)

  # no docs for private functions
  defp do_size(filename_or_blob, mode \\ nil)
end
```

@rrrene

5Minds IT-SOLUTIONS

summary

```elixir
defmodule MyModule do
  @typedoc "Ecto.Query metadata fields (stored in cache)"
  @type query_meta :: %{sources: tuple, preloads: term, select: map}

  @doc "A callback executed when the supervisor starts"
  @callback init(config :: Keyword.t()) :: {:ok, Keyword.t()} | :ignore
end
```

(͡° ͜ʖ ͡°)ง

But what is the problem?

@rrrene

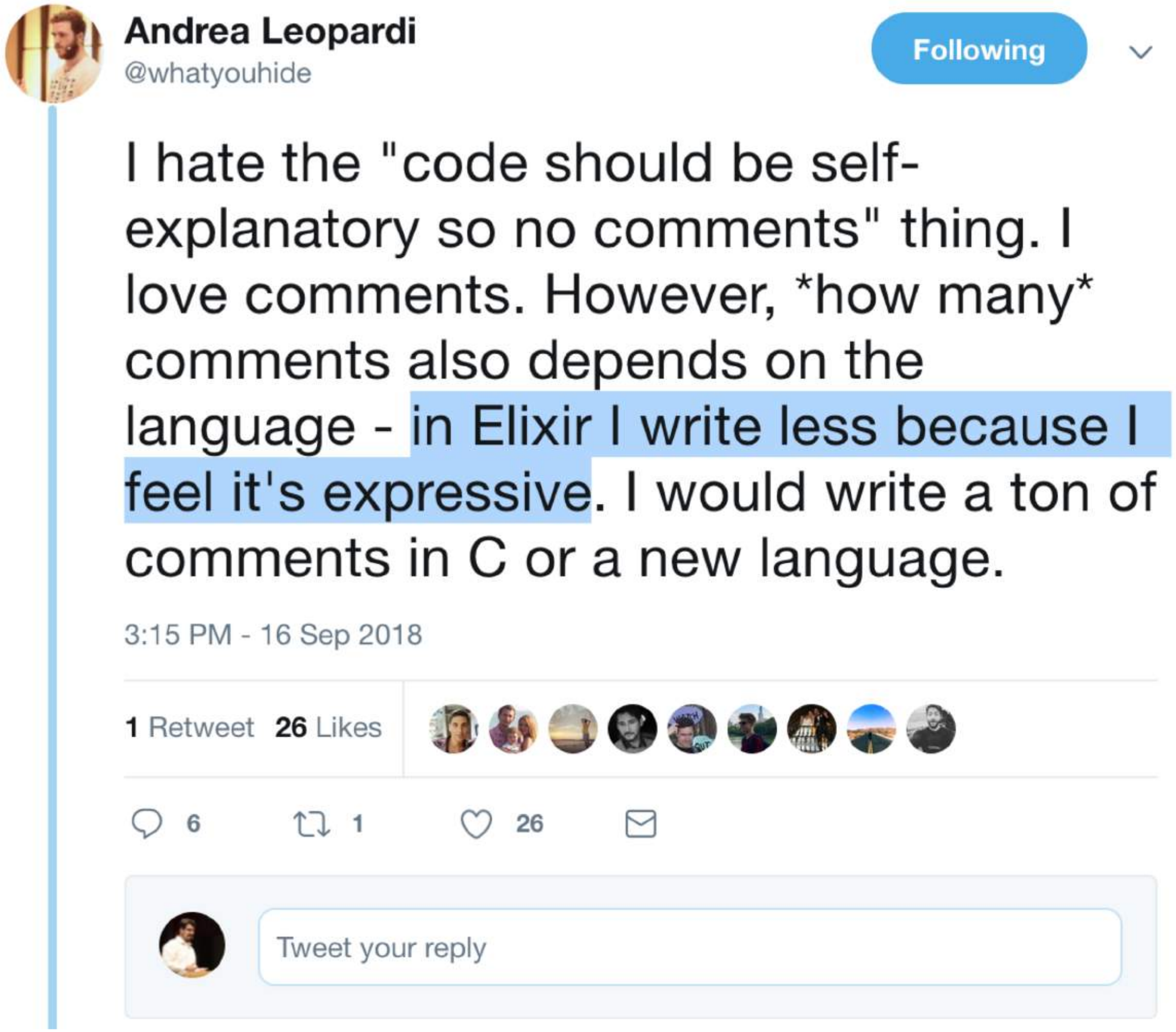
„good code is its own documentation“

(myself in my early twenties)

versus

„people are not code parsers“

(Zach Holman)



Andrea Leopardi
@whatyouhide

Following

I hate the "code should be self-explanatory so no comments" thing. I love comments. However, *how many* comments also depends on the language - in Elixir I write less because I feel it's expressive. I would write a ton of comments in C or a new language.

3:15 PM - 16 Sep 2018

1 Retweet 26 Likes

6 1 26

Tweet your reply

@rrrene

5Minds IT-SOLUTIONS

Tooling helps!

because, there must be tools, right?

Dialyzer

build passing code climate 3.8

Dialyzer

coverage 99% dependencies up-to-date

„There are 0 lines of documentation.“

or

„65.7% documented“

Look, here are the facts.

Designing Inch
Creating a more opinionated tool.

Making up the rules.

```
defmodule :... do
  # Let's look at all code objects ...
  @code_objects ~w(
    modules
    functions
    parameters
  )

  # ... and assign assign a score to them.
  @scores 0..100
end
```

5Minds IT-SOLUTIONS

Code.get_docs/2

```
Code.get_docs(Plug.Builder, :all)
```

```
[
  docs: [
    {{:__before_compile__, 1}, 125, :defmacro, [{:env, [], nil}], false},
    {{:__using__, 1}, 101, :defmacro, [{:opts, [], nil}], false},
    {{:compile, 3}, 156, :def,
      [{:env, [], nil}, {:pipeline, [], nil}, {:builder_opts, [], nil}],
        "Compiles a plug pipeline ..."},
    {{:plug, 2}, 137, :defmacro,
      [{:plug, [], nil}, {:\\, [], [{:opts, [], nil}, []]}],
        "A macro that stores a new plug. ..."}
  ],
  moduledoc: {2, "Conveniences for building plugs ..."},
  callback_docs: [],
  type_docs: [{{:plug, 0}, 99, :type, nil}]
]
```

```
[
  docs: [
    {{:__before_compile__, 1}, 125, :defmacro, [{:env, [], nil}], false},
    {{:__using__, 1}, 101, :defmacro, [{:opts, [], nil}], false},
    {{:compile, 3}, 156, :def,
      [{:env, [], nil}, {:pipeline, [], nil}, {:builder_opts, [], nil}],
        "Compiles a plug pipeline ..."},
    {{:plug, 2}, 137, :defmacro,
      [{:plug, [], nil}, {:\\, [], [{:opts, [], nil}, []]}],
        "A macro that stores a new plug. ..."}
  ],
  moduledoc: {2, "Conveniences for building plugs ..."},
  callback_docs: [],
  type_docs: [{{:plug, 0}, 99, :type, nil}]
]
```

```
[
  docs: [
    {{:__before_compile__, 1}, 125, :defmacro, [{:env, [], nil}], false},
    {{:__using__, 1}, 101, :defmacro, [{:opts, [], nil}], false},
    {{:compile, 3}, 156, :def,
      [{:env, [], nil}, {:pipeline, [], nil}, {:builder_opts, [], nil}],
        "Compiles a plug pipeline ..."},
    {{:plug, 2}, 137, :defmacro,
      [{:plug, [], nil}, {:\\, [], [{:opts, [], nil}, []]}],
        "A macro that stores a new plug. ..."}
  ],
  moduledoc: {2, "Conveniences for building plugs ..."},
  callback_docs: [],
  type_docs: [{{:plug, 0}, 99, :type, nil}]
]
```

@rrrene

5Minds IT-SOLUTIONS

```
[
  docs: [
    {{:__before_compile__, 1}, 125, :defmacro, [{:env, [], nil}], false},
    {{:__using__, 1}, 101, :defmacro, [{:opts, [], nil}], false},
    {{:compile, 3}, 156, :def,
      [{:env, [], nil}, {:pipeline, [], nil}, {:builder_opts, [], nil}],
        "Compiles a plug pipeline ..."},
    {{:plug, 2}, 137, :defmacro,
      [{:plug, [], nil}, {:\\, [], [{:opts, [], nil}, []]}],
        "A macro that stores a new plug. ..."}
  ],
  moduledoc: {2, "Conveniences for building plugs ..."},
  callback_docs: [],
  type_docs: [{{:plug, 0}, 99, :type, nil}]
]
```

```
[
  docs: [
    {{:__before_compile__, 1}, 125, :defmacro, [{:env, [], nil}], false},
    {{:__using__, 1}, 101, :defmacro, [{:opts, [], nil}], false},
    {{:compile, 3}, 156, :def,
      [{:env, [], nil}, {:pipeline, [], nil}, {:builder_opts, [], nil}],
        "Compiles a plug pipeline ..."},
    {{:plug, 2}, 137, :defmacro,
      [{:plug, [], nil}, {:\\, [], [{:opts, [], nil}, []]}],
        "A macro that stores a new plug. ..."}
  ],
  moduledoc: {2, "Conveniences for building plugs ..."},
  callback_docs: [],
  type_docs: [{{:plug, 0}, 99, :type, nil}]
]
```

Code.get_docs/2

Cod~~e~~ ~~...~~s/2

deprecated

@rrrene

5Minds IT-SOLUTIONS

Then came Elixir 1.7 ... i.e. EEP 48!!!

What is EEP 48?

- Storage format for documentation
- Compatible across all BEAM languages (Erlang, Elixir, LFE, …)
- Allows for individual „styles“ in each language
- Will allow us to use stuff (tools, libraries, etc.) across languages more easily
- Less friction! Yay!

Code.fetch_docs/1

Code.fetch_docs(Plug.Builder)

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@rrrene

5Minds IT-SOLUTIONS

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@rrrene

5Minds IT-SOLUTIONS

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@rrrene

5Minds IT-SOLUTIONS

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@rrrene

5Minds IT-SOLUTIONS

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@rrrene

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@rrrene

@doc "Compiles a plug pipeline ..."

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@doc "Compiles a plug pipeline ..."

@doc author: "rrrene"

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{author: “rrrene"}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_compile__(env)"],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@doc "Compiles a plug pipeline ..."

@doc author: "rrrene"

@rrrene

5Minds IT-SOLUTIONS

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_com ],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@doc "Compiles a plug pipeline ..."

@doc author: "rrrene"

@doc false

```
{:docs_v1, 2, :elixir, "text/markdown",
  %{"en" => "Conveniences for building plugs ..."}, %{},
  [
    {{:function, :compile, 3}, 156, ["compile(env, pipeline, builder_opts)"],
      %{"en" => "Compiles a plug pipeline ..."}, %{}},
    {{:macro, :__before_compile__, 1}, 125, ["__before_com                ],
      :hidden, %{}},
    {{:macro, :__using__, 1}, 101, ["__using__(opts)"], :hidden, %{}},
    {{:macro, :plug, 2}, 137, ["plug(plug, opts \\\\ [])"],
      %{"en" => "A macro that stores a new plug ..."}, %{defaults: 1}},
    {{:type, :plug, 0}, 99, [], :none, %{}}
  ]}
```

@doc "Compiles a plug pipeline ..."

@doc author: "rrrene"

@doc false

no @doc attribute

```
# Where was I? ... Ah, yes, how to measure documentation!
module
|> Code.fetch_docs()
|> code_objects()
|> attach_roles()
|> assign_scores()
```

```
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)
```

doc string present

```
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)
```

doc string present

```
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)
```

code example present

@rrrene

5Minds IT-SOLUTIONS

doc string present

parameter mentioned

```
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)
```

code example present

@rrrene

5Minds IT-SOLUTIONS

doc string present

parameter mentioned

```
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)
```

code example present

parameter not mentioned

{"with_docstring", _string}

{"with_function_parameter_mention",
  {_name, _count}}

```
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""

def size(filename_or_blob, mode \\ nil)
```

{"with_code_example", _string}

{"without_function_parameter_mention",
  {_name, _count}}

@rrrene

5Minds IT-SOLUTIONS

```
def score({"with_docstring", _}), do: 50

def score({"with_code_example", _}), do: 10

def score({"with_function_parameter_mention", {_name, count}}) do
  div(40, count)
end

def score(_), do: 0

iex> Enum.reduce(code_object.roles, 0, &(score(&1) + &2))
80
```

@rrene

So, are we done?

it is more important to document … top-level functions than internal ones

5Minds IT-SOLUTIONS

it is more important to document …

top-level functions than internal ones

functions with many parameters

it is more important to document …
top-level functions than internal ones
functions with many parameters
modules containing functions

```
defmodule App do
  def register_user(attributes)
end

defmodule App.Models do
end

defmodule App.Models.User do
  def register(name, email, password)
end
```

```elixir
defmodule App do
  def register_user(attributes)
end

defmodule App.Models do
end

defmodule App.Models.User do
  def register(name, email, password)
end
```

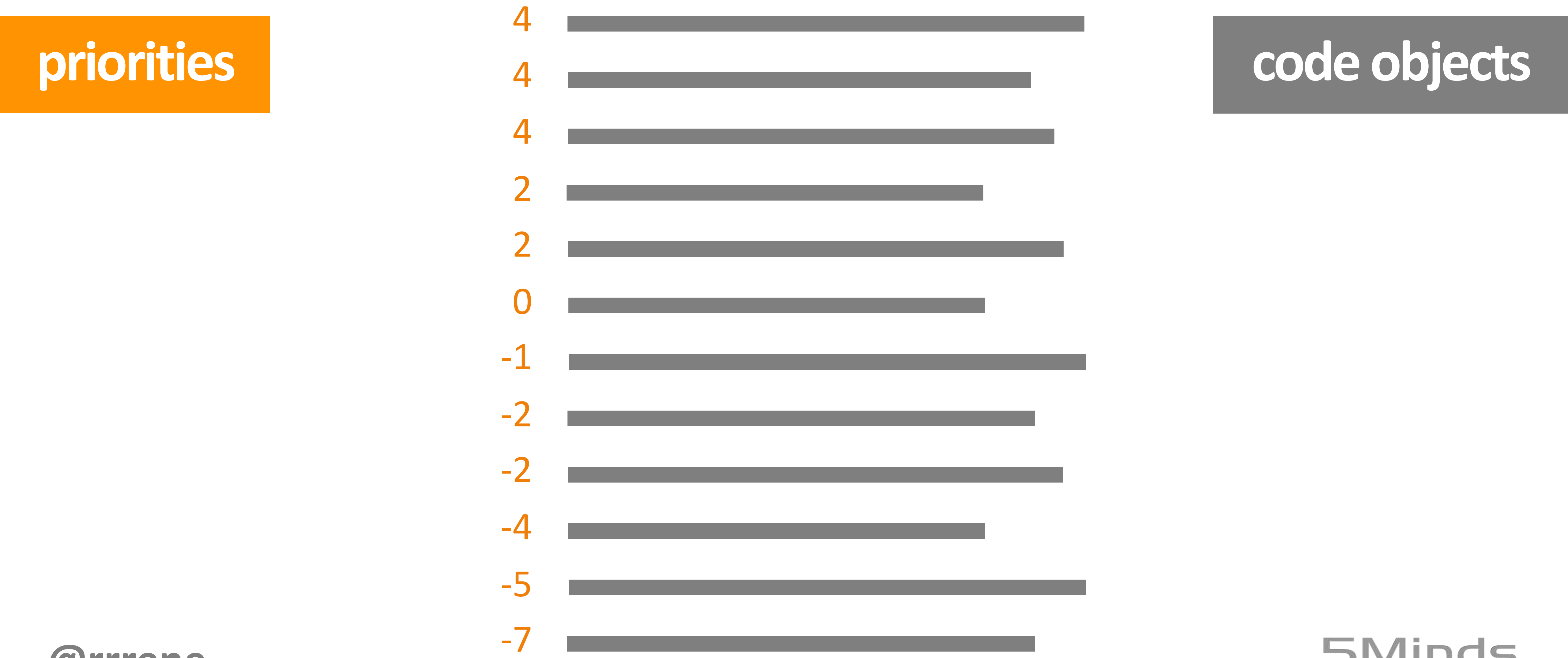
modules

```
defmodule App do
  def register_user(attributes)
end

defmodule App.Models do
end

defmodule App.Models.User do
  def register(name, email, password)
end
```

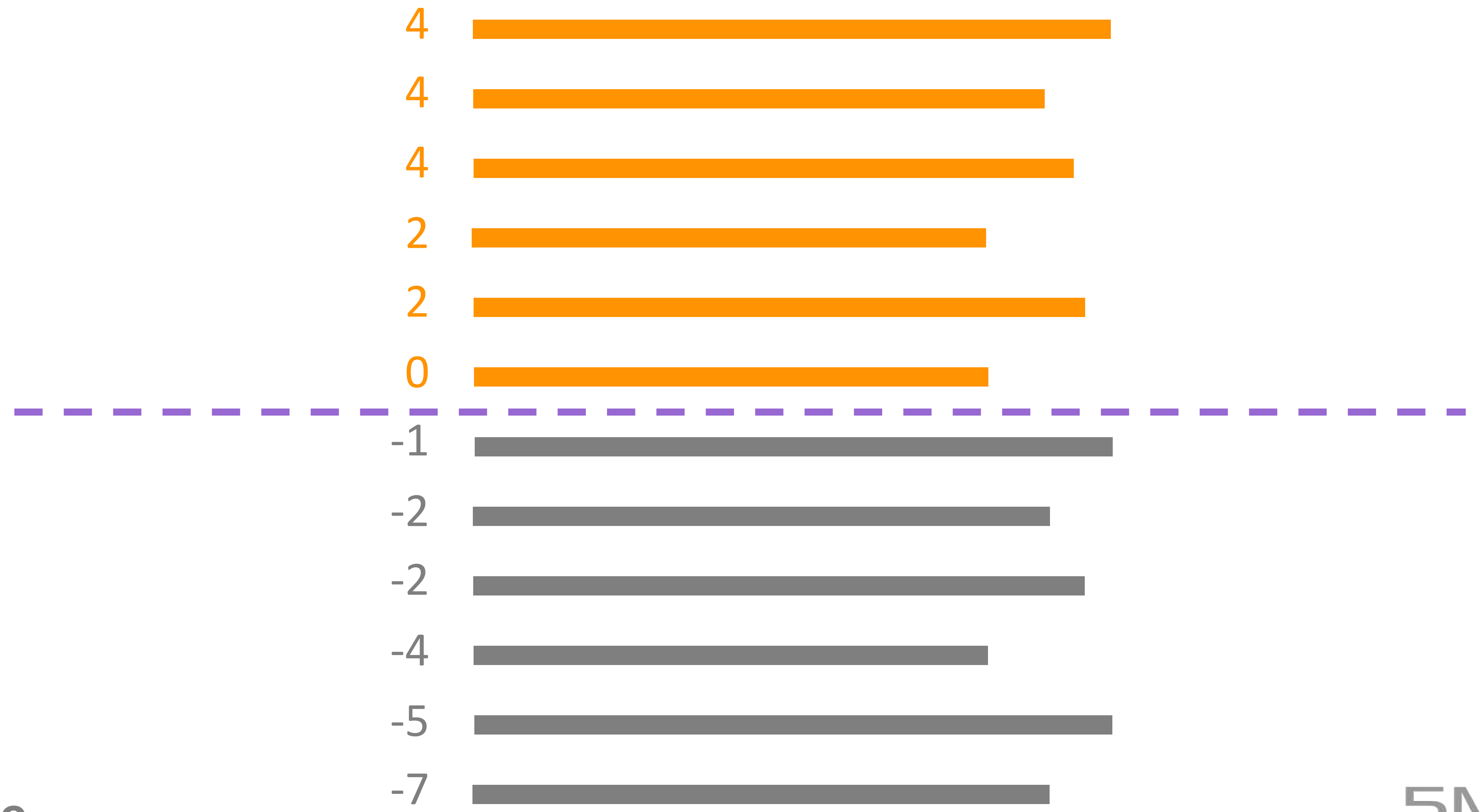
functions

it is more important to document …
top-level functions than internal ones
functions with many parameters
modules containing functions
…

code objects ordered by priority



code objects ordered by priority



```
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)

Score: ?
Priority: ?
```

```elixir
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)
```

Score: 80/100

Priority: ↗

```
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)

Score: 80/100
Priority: ↗
```

```elixir
@doc """
Returns the size of a given `filename_or_blob`.

    iex> MyModule.size(filename)
    4096
"""
def size(filename_or_blob, mode \\ nil)
```

Grade: B

Priority: ↗

```
iex> InchEx.GradeList.all()
A – Really good
B – Proper documentation found
C – Please take a look
U – Undocumented
```

```
iex> CLI.main([])
```

```
$ mix inch

# Proper documentation present
┃
┃ 80  4 Phoenix.Token.sign/4 (lib/phoenix/token.ex:94)
┃ 70  2 Mix.Phoenix.inflect/1 (lib/mix/phoenix.ex:57)
┃ 70 -1 Phoenix.Endpoint.Supervisor.server?/2 (lib/phoenix/endpoint/supervisor.ex:103)

# Undocumented
┃
┃  0  2 Phoenix.Param.Any (lib/phoenix/param.ex:82)
┃  0  1 Phoenix.Param.Map (lib/phoenix/param.ex:75)
┃  0 -1 Mix.Phoenix.Schema.valid?/1 (lib/mix/phoenix/schema.ex:41)

You might want to look at these files:

┃  lib/phoenix/controller.ex
┃  lib/phoenix/test/conn_test.ex

Grade distribution (undocumented, C, B, A): █ ▁ ▇ ▂
```

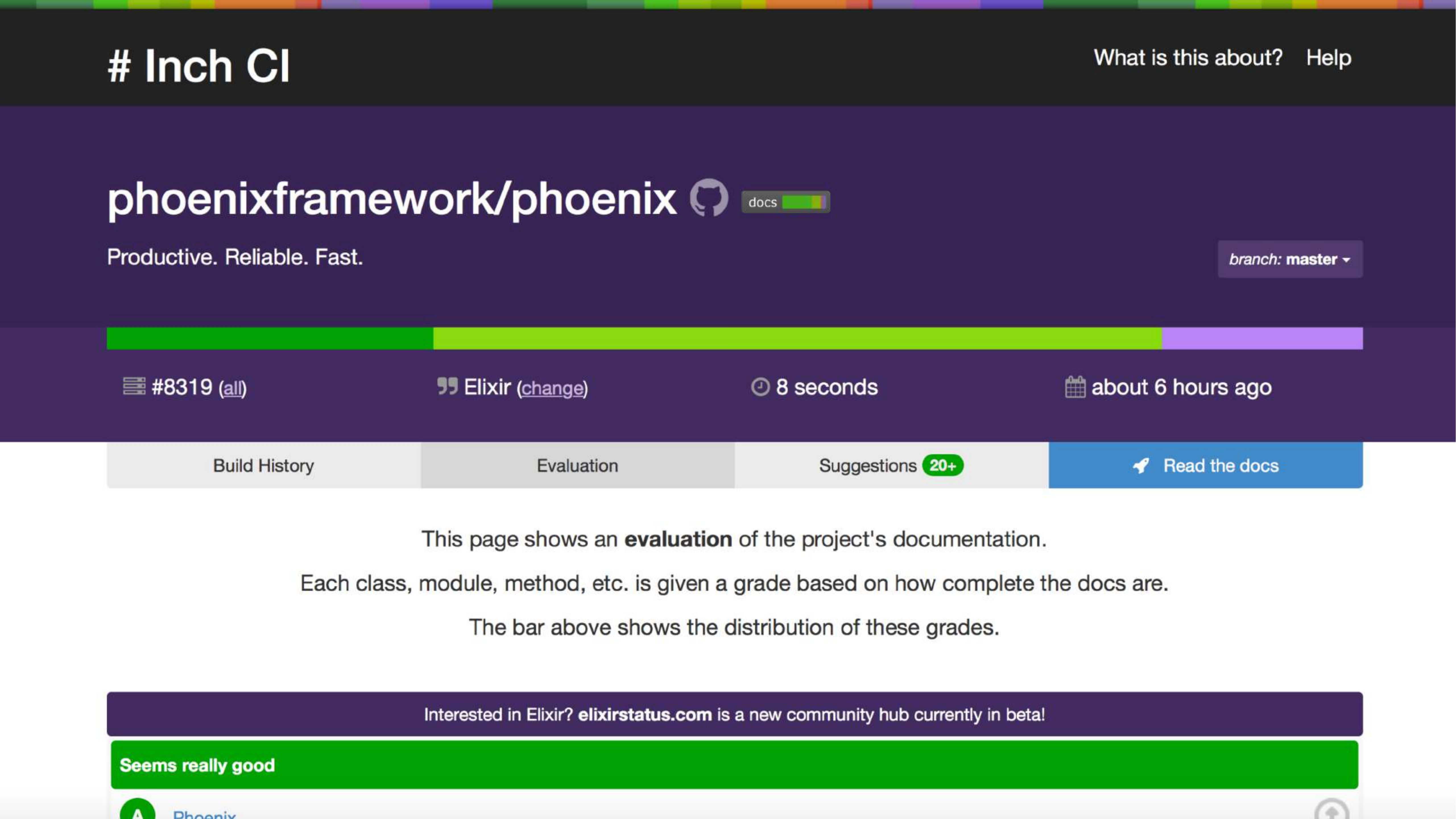
```
$ mix inch

# Proper documentation present
┃
┃ [B] → Phoenix.Token.sign/4 (lib/phoenix/token.ex:94)
┃ [B] → Mix.Phoenix.inflect/1 (lib/mix/phoenix.ex:57)
┃ [B] ↓ Phoenix.Endpoint.Supervisor.server?/2 (lib/phoenix/endpoint/supervisor.ex:103)

# Undocumented
┃
┃ [U] → Phoenix.Param.Any (lib/phoenix/param.ex:82)
┃ [U] → Phoenix.Param.Map (lib/phoenix/param.ex:75)
┃ [U] ↓ Mix.Phoenix.Schema.valid?/1 (lib/mix/phoenix/schema.ex:41)

You might want to look at these files:

┃  lib/phoenix/controller.ex
┃  lib/phoenix/test/conn_test.ex

Grade distribution (undocumented, C, B, A): █ ▁ ▇ ▂
```

```
$ mix inch

# Proper documentation present
┃
┃ [B] → Phoenix.Token.sign/4 (lib/phoenix/token.ex:94)
┃ [B] → Mix.Phoenix.inflect/1 (lib/mix/phoenix.ex:57)
┃ [B] ↓ Phoenix.Endpoint.Supervisor.server?/2 (lib/phoenix/endpoint/supervisor.ex:103)

# Undocumented
┃
┃ [U] → Phoenix.Param.Any (lib/phoenix/param.ex:82)
┃ [U] → Phoenix.Param.Map (lib/phoenix/param.ex:75)
┃ [U] ↓ Mix.Phoenix.Schema.valid?/1 (lib/mix/phoenix/schema.ex:41)

You might want to look at these files:

┃   lib/phoenix/controller.ex
┃   lib/phoenix/test/conn_test.ex

Grade distribution (undocumented, C, B, A): █ ▁ █ ▂
```

no overall grade!

```
iex> CLI.main([])
iex> CLI.main([])
```

but how to get people excited about this?



Inch CI

What is this about? Help

phoenixframework/phoenix docs

Productive. Reliable. Fast.

branch: **master**

#8319 (all) Elixir (change) 8 seconds about 6 hours ago

Build History | Evaluation | Suggestions 20+ | Read the docs

This page shows an **evaluation** of the project's documentation.

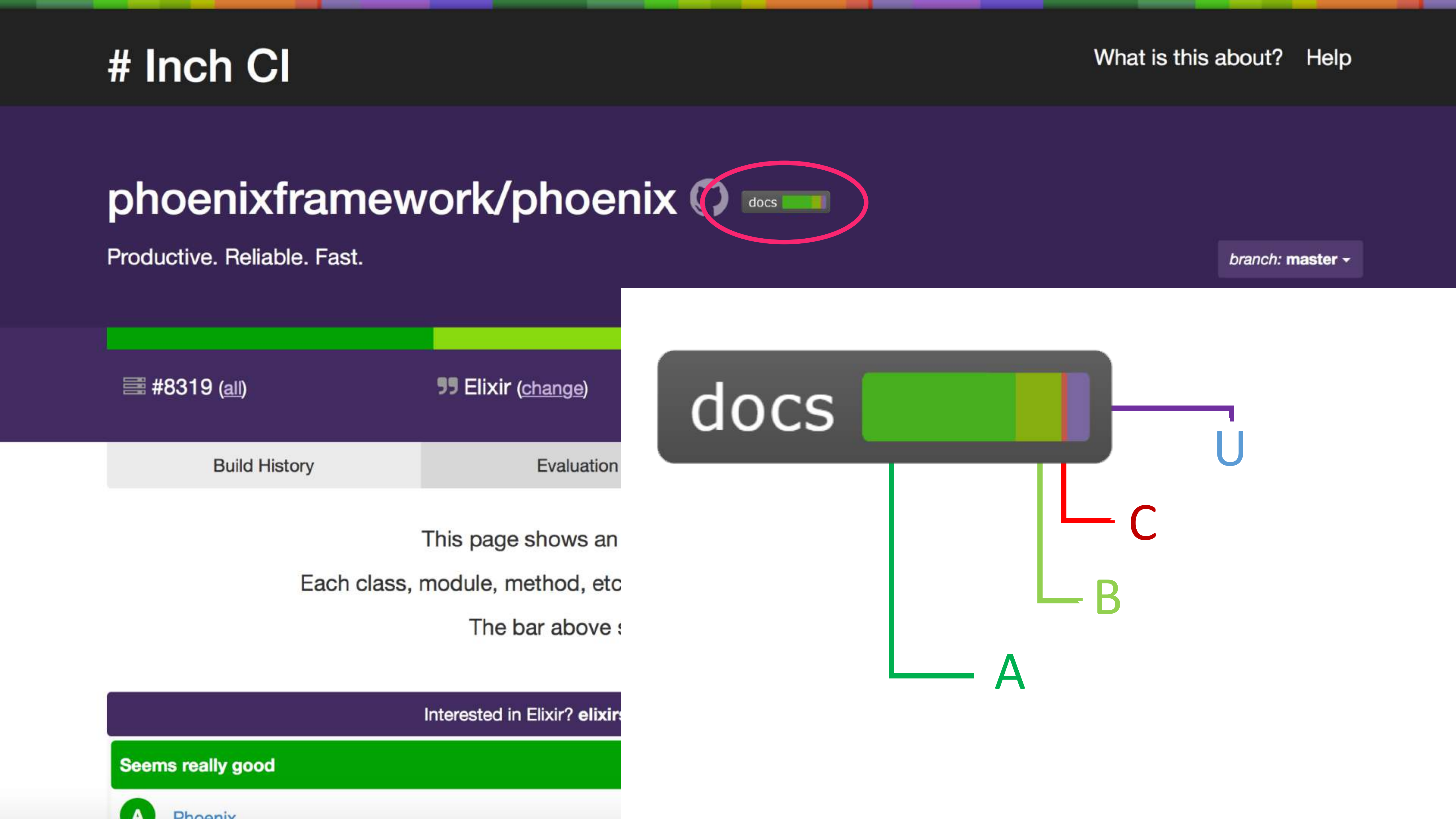
Each class, module, method, etc. is given a grade based on how complete the docs are.

The bar above shows the distribution of these grades.

Interested in Elixir? **elixirstatus.com** is a new community hub currently in beta!

Seems really good

A Phoenix



Inch CI

What is this about? Help

phoenixframework/phoenix

Productive. Reliable. Fast.

branch: master

#8319 (all)

Elixir (change)

Build History

Evaluation

This page shows an

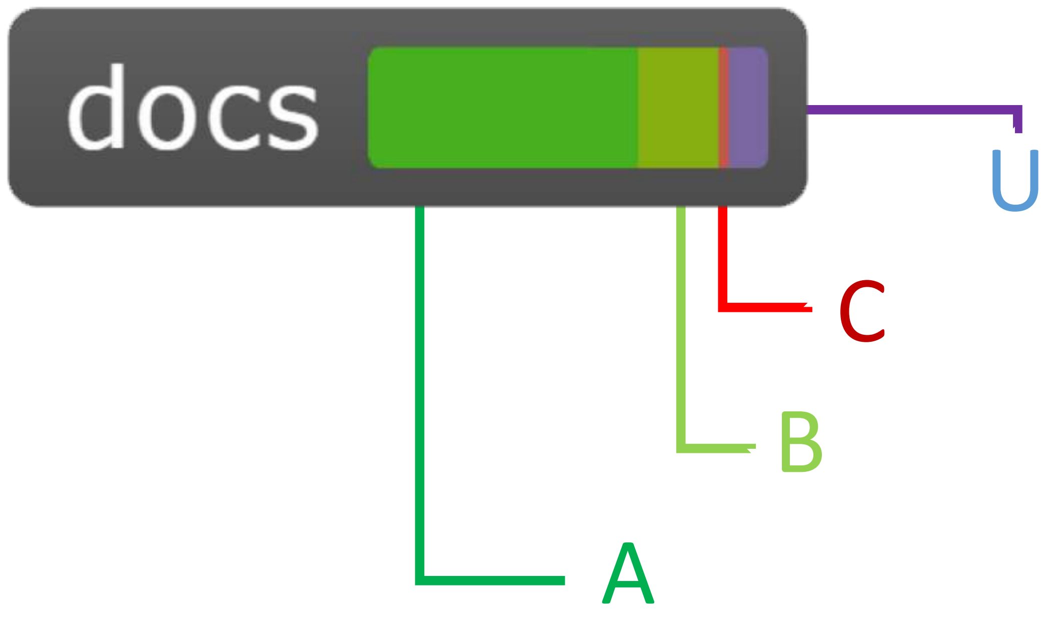
Each class, module, method, etc

The bar above s

Interested in Elixir? elixir

Seems really good

Phoenix



Lessons learned building Inch

#1
Software is people business

#2
Approach Open Source like Fight Club

#3
Show, don't tell

@rrrene

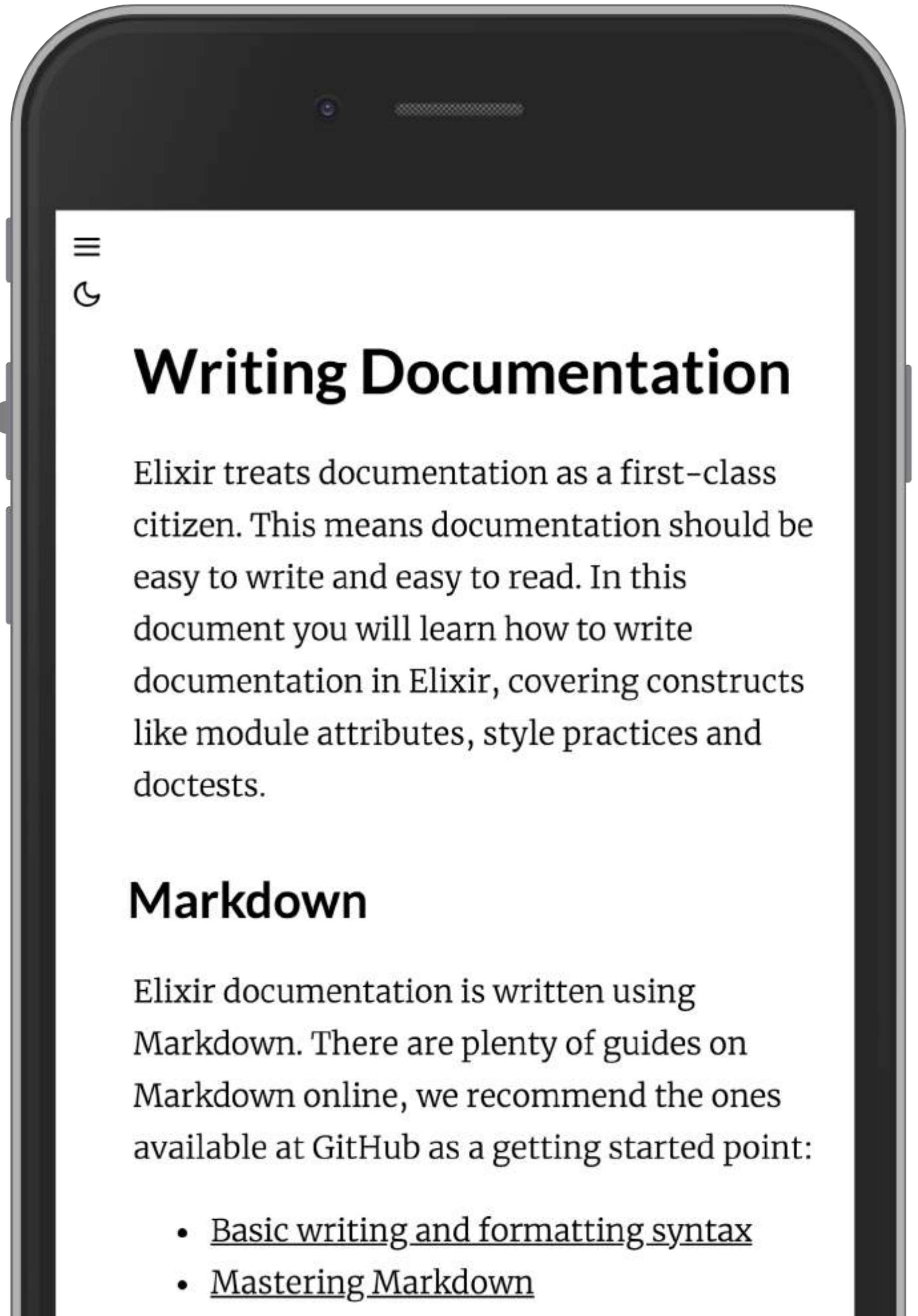
5Minds
IT-SOLUTIONS

Further reading

Page “Writing Documentation"

can be found inside the official Elixir docs

https://hexdocs.pm/elixir/writing-documentation.html



Writing Documentation

Elixir treats documentation as a first-class citizen. This means documentation should be easy to write and easy to read. In this document you will learn how to write documentation in Elixir, covering constructs like module attributes, style practices and doctests.

Markdown

Elixir documentation is written using Markdown. There are plenty of guides on Markdown online, we recommend the ones available at GitHub as a getting started point:

- Basic writing and formatting syntax
- Mastering Markdown

@rrrene

Inch

HOW ELIXIR 1.7 CHANGED THE RULES

FOR DOCUMENTATION ANALYSIS

René Föhring, Berlin, 2018

@rrrene

5Minds IT-SOLUTIONS