Getting Help
|> Ways to Contribute

Topics

1. Where to start learning about Elixir
2. Tools to use during project development
3. Where to get help when writing code
4. How to contribute back

Elixir: Getting Started

Elixir Homepage (elixir-lang.org)

- Always up-to-date
- Install, Getting Started, Learning
- Crash course for Erlang developers
- Links to the community
- Code Editor support

Books

- *Programming Elixir 1.2*
- *Elixir in Action*
- *Metaprogramming Elixir*
- *Programming Phoenix*
- *The Little Elixir & OTP Guidebook*
- redfour.io

Tutorials, Screencasts, Podcasts...

- Elixir School
- Elixir Sips, LearnElixir.tv
- Elixir Fountain

Down to Erlang

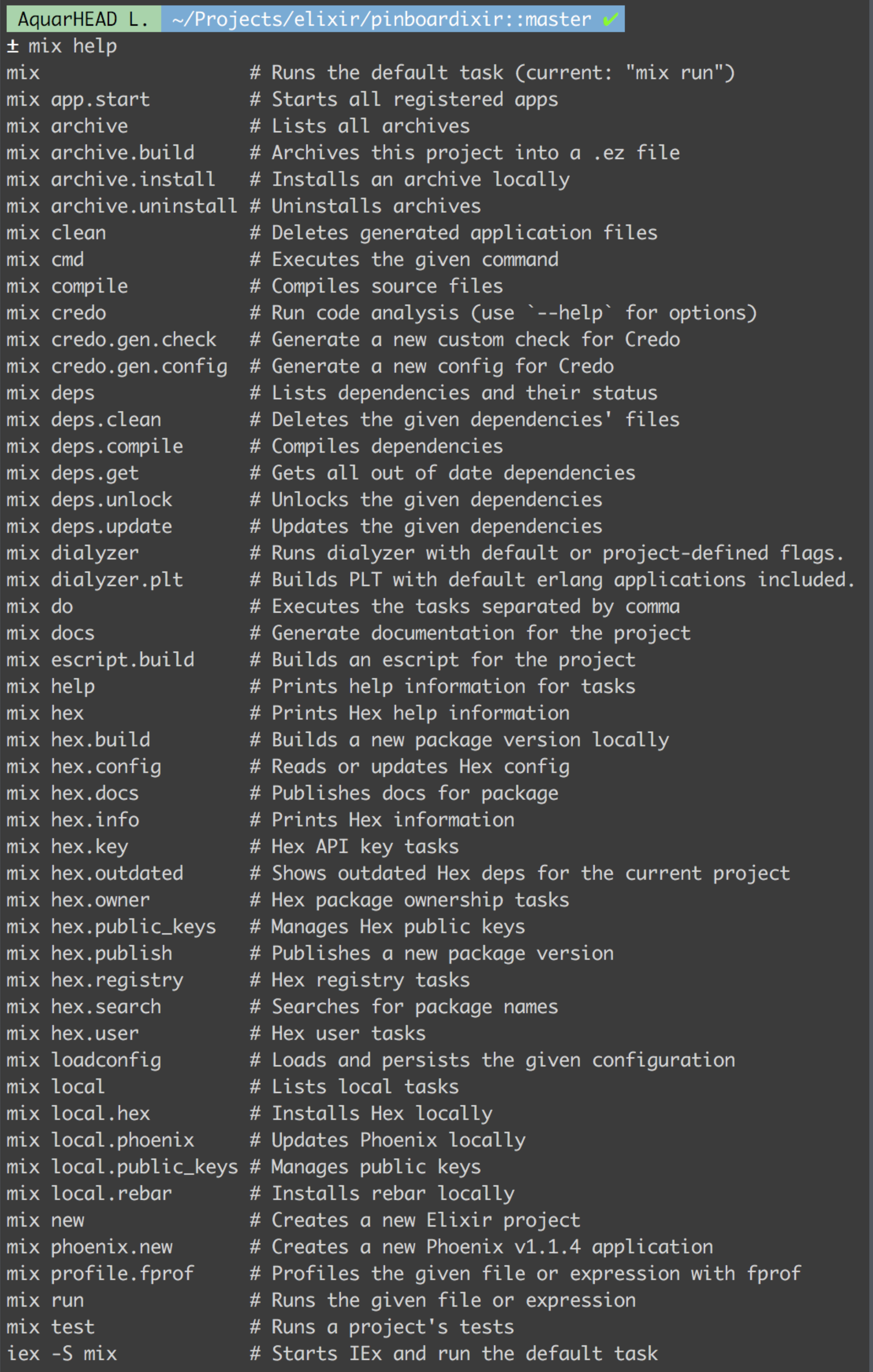
- *Programming Erlang & Erlang Programming*
- *Learn You Some Erlang for Great Good*
- *Erlang and OTP in Action*
- *Designing for Scalability with Erlang/OTP*

Developing an Elixir Project

To Elixir, You `mix`

- From Ruby? `mix` ~= `gem` + `bundle` + `rake`
- Create, compile, run, test your application
- Manage dependencies
- Generate docs, publish both app and docs
- Credo, Dialyzer, etc..

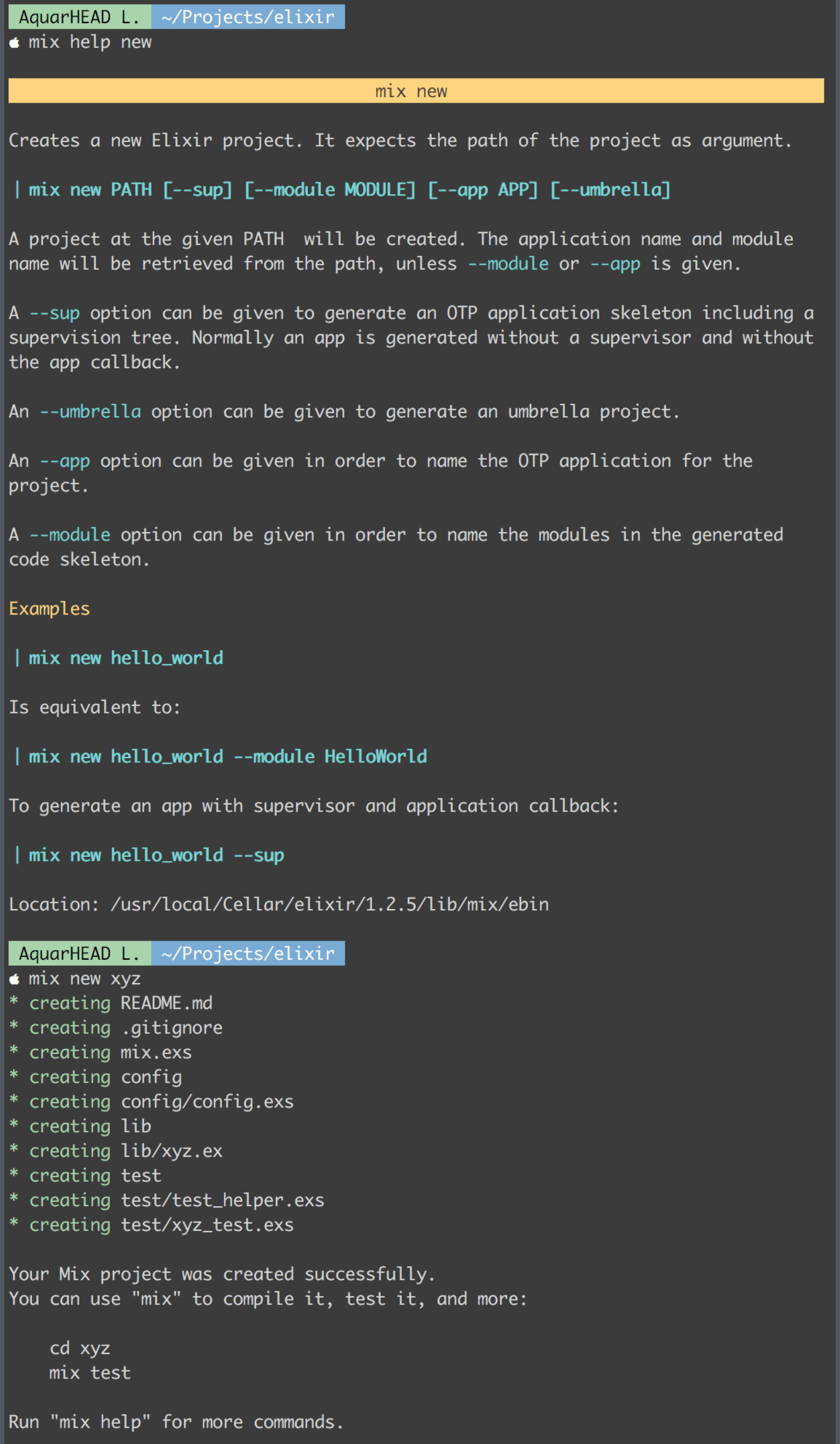
Lou Xun aquarhead@ela.build / Ela Workshop



```
AquarHEAD L.  ~/Projects/elixir/pinboardixir::master ✔
± mix help
mix                      # Runs the default task (current: "mix run")
mix app.start            # Starts all registered apps
mix archive              # Lists all archives
mix archive.build        # Archives this project into a .ez file
mix archive.install      # Installs an archive locally
mix archive.uninstall    # Uninstalls archives
mix clean                # Deletes generated application files
mix cmd                  # Executes the given command
mix compile              # Compiles source files
mix credo                # Run code analysis (use `--help` for options)
mix credo.gen.check      # Generate a new custom check for Credo
mix credo.gen.config     # Generate a new config for Credo
mix deps                 # Lists dependencies and their status
mix deps.clean           # Deletes the given dependencies' files
mix deps.compile         # Compiles dependencies
mix deps.get             # Gets all out of date dependencies
mix deps.unlock          # Unlocks the given dependencies
mix deps.update          # Updates the given dependencies
mix dialyzer             # Runs dialyzer with default or project-defined flags.
mix dialyzer.plt         # Builds PLT with default erlang applications included.
mix do                   # Executes the tasks separated by comma
mix docs                 # Generate documentation for the project
mix escript.build        # Builds an escript for the project
mix help                 # Prints help information for tasks
mix hex                  # Prints Hex help information
mix hex.build            # Builds a new package version locally
mix hex.config           # Reads or updates Hex config
mix hex.docs             # Publishes docs for package
mix hex.info             # Prints Hex information
mix hex.key              # Hex API key tasks
mix hex.outdated         # Shows outdated Hex deps for the current project
mix hex.owner            # Hex package ownership tasks
mix hex.public_keys      # Manages Hex public keys
mix hex.publish          # Publishes a new package version
mix hex.registry         # Hex registry tasks
mix hex.search           # Searches for package names
mix hex.user             # Hex user tasks
mix loadconfig           # Loads and persists the given configuration
mix local                # Lists local tasks
mix local.hex            # Installs Hex locally
mix local.phoenix        # Updates Phoenix locally
mix local.public_keys    # Manages public keys
mix local.rebar          # Installs rebar locally
mix new                  # Creates a new Elixir project
mix phoenix.new          # Creates a new Phoenix v1.1.4 application
mix profile.fprof        # Profiles the given file or expression with fprof
mix run                  # Runs the given file or expression
mix test                 # Runs a project's tests
iex -S mix               # Starts IEx and run the default task
```

mix new

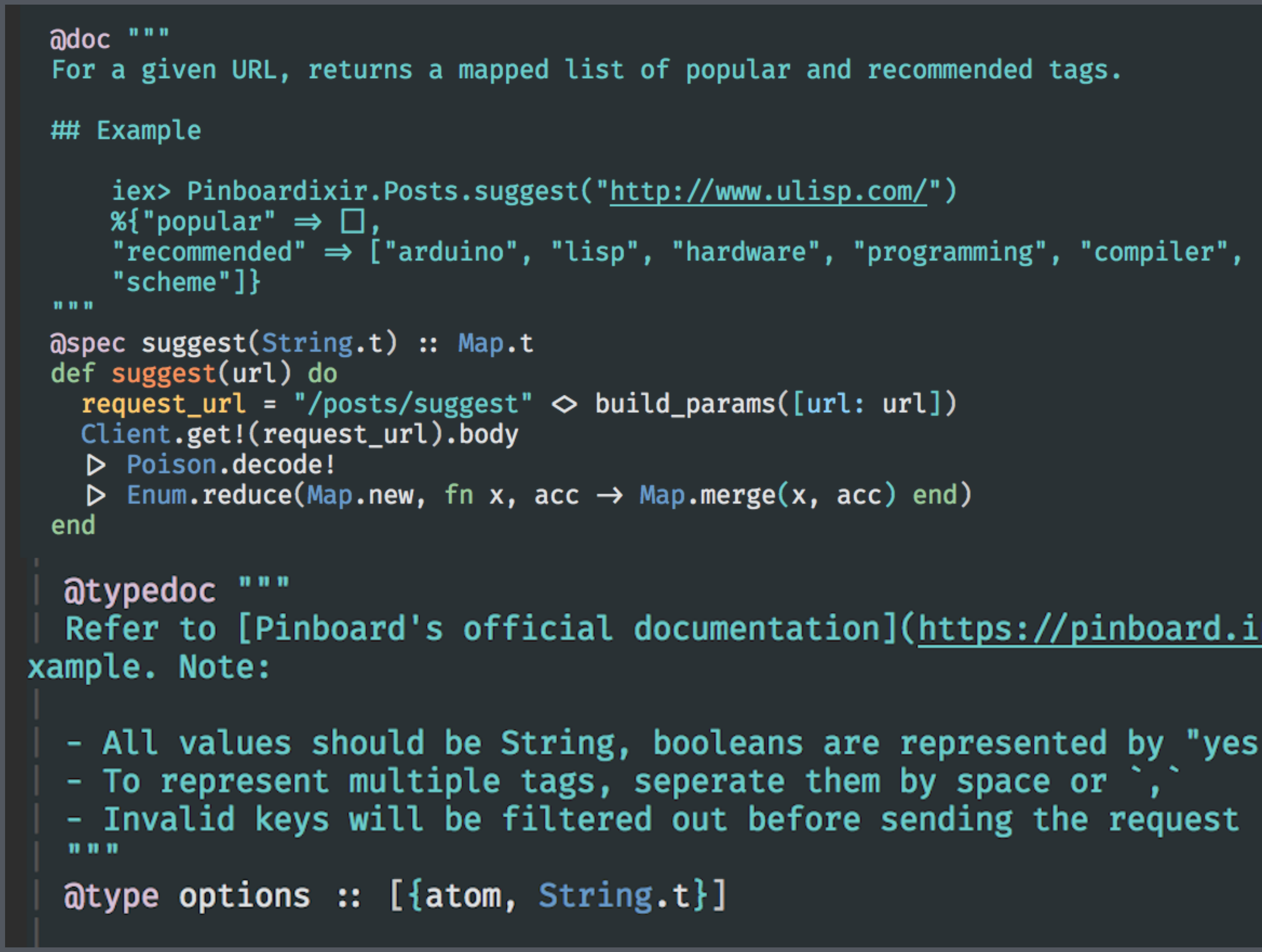
- Provides sane defaults (project structure)
- Encourages "best practices"
- Umbrella project



```
AquarHEAD L.  ~/Projects/elixir
 mix help new

                                   mix new

Creates a new Elixir project. It expects the path of the project as argument.

 | mix new PATH [--sup] [--module MODULE] [--app APP] [--umbrella]

A project at the given PATH  will be created. The application name and module
name will be retrieved from the path, unless --module or --app is given.

A --sup option can be given to generate an OTP application skeleton including a
supervision tree. Normally an app is generated without a supervisor and without
the app callback.

An --umbrella option can be given to generate an umbrella project.

An --app option can be given in order to name the OTP application for the
project.

A --module option can be given in order to name the modules in the generated
code skeleton.

Examples

 | mix new hello_world

Is equivalent to:

 | mix new hello_world --module HelloWorld

To generate an app with supervisor and application callback:

 | mix new hello_world --sup

Location: /usr/local/Cellar/elixir/1.2.5/lib/mix/ebin

AquarHEAD L.  ~/Projects/elixir
 mix new xyz
* creating README.md
* creating .gitignore
* creating mix.exs
* creating config
* creating config/config.exs
* creating lib
* creating lib/xyz.ex
* creating test
* creating test/test_helper.exs
* creating test/xyz_test.exs

Your Mix project was created successfully.
You can use "mix" to compile it, test it, and more:

    cd xyz
    mix test

Run "mix help" for more commands.
```

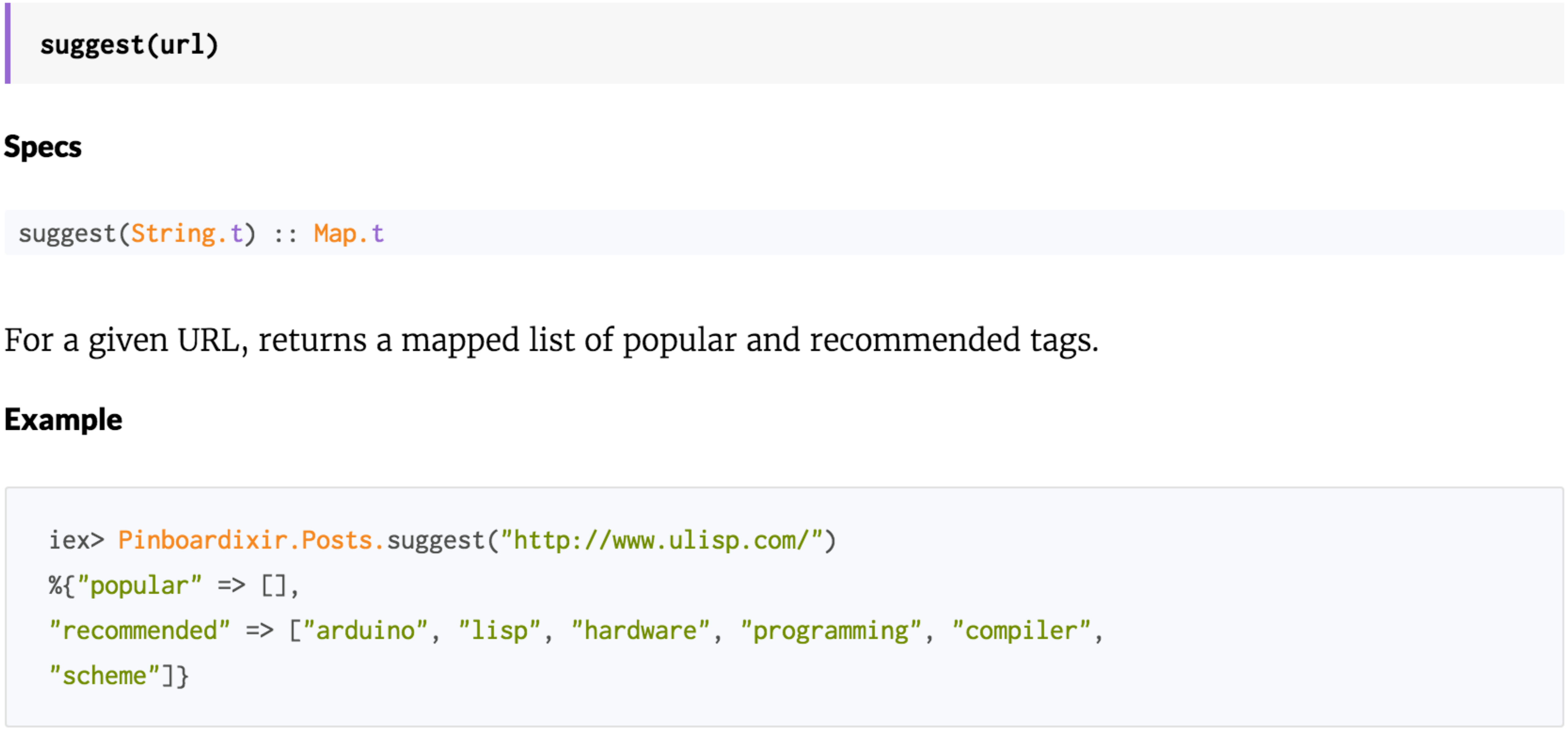
Documentation and Typespec

- `@moduledoc (false)`
- `@doc`
- Use Markdown
- Include some examples and get them tested as well!
- `mix docs`
- `@typedoc`, `@type`, `@typep`
- `mix dialyzer`



```elixir
@doc """
For a given URL, returns a mapped list of popular and recommended tags.

## Example

    iex> Pinboardixir.Posts.suggest("http://www.ulisp.com/")
    %{"popular" => [],
    "recommended" => ["arduino", "lisp", "hardware", "programming", "compiler",
    "scheme"]}
"""
@spec suggest(String.t) :: Map.t
def suggest(url) do
  request_url = "/posts/suggest" <> build_params([url: url])
  Client.get!(request_url).body
  |> Poison.decode!
  |> Enum.reduce(Map.new, fn x, acc -> Map.merge(x, acc) end)
end
```

```elixir
 @typedoc """
 Refer to [Pinboard's official documentation](https://pinboard.i
xample. Note:

 - All values should be String, booleans are represented by "yes
 - To represent multiple tags, seperate them by space or `,`
 - Invalid keys will be filtered out before sending the request
 """
 @type options :: [{atom, String.t}]
```



```
suggest(url)
```

Specs

```
suggest(String.t) :: Map.t
```

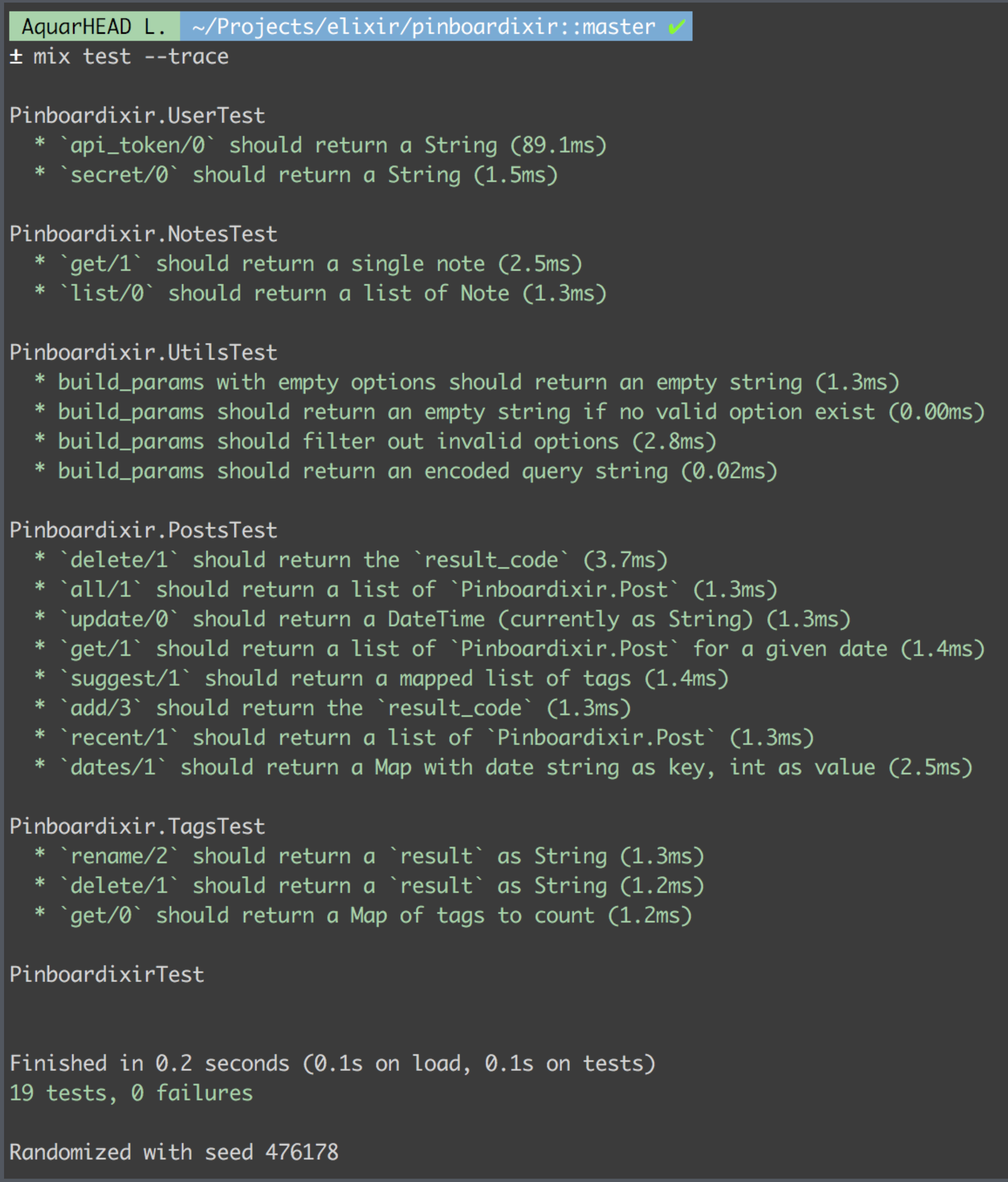
For a given URL, returns a mapped list of popular and recommended tags.

Example

```
iex> Pinboardixir.Posts.suggest("http://www.ulisp.com/")
%{"popular" => [],
"recommended" => ["arduino", "lisp", "hardware", "programming", "compiler",
"scheme"]}
```

Testing

- `ExUnit.Case`, `async`
- Filter tests based on tags
- Seed random numbers



```
AquarHEAD L.  ~/Projects/elixir/pinboardixir::master ✔
± mix test --trace

Pinboardixir.UserTest
  * `api_token/0` should return a String (89.1ms)
  * `secret/0` should return a String (1.5ms)

Pinboardixir.NotesTest
  * `get/1` should return a single note (2.5ms)
  * `list/0` should return a list of Note (1.3ms)

Pinboardixir.UtilsTest
  * build_params with empty options should return an empty string (1.3ms)
  * build_params should return an empty string if no valid option exist (0.00ms)
  * build_params should filter out invalid options (2.8ms)
  * build_params should return an encoded query string (0.02ms)

Pinboardixir.PostsTest
  * `delete/1` should return the `result_code` (3.7ms)
  * `all/1` should return a list of `Pinboardixir.Post` (1.3ms)
  * `update/0` should return a DateTime (currently as String) (1.3ms)
  * `get/1` should return a list of `Pinboardixir.Post` for a given date (1.4ms)
  * `suggest/1` should return a mapped list of tags (1.4ms)
  * `add/3` should return the `result_code` (1.3ms)
  * `recent/1` should return a list of `Pinboardixir.Post` (1.3ms)
  * `dates/1` should return a Map with date string as key, int as value (2.5ms)

Pinboardixir.TagsTest
  * `rename/2` should return a `result` as String (1.3ms)
  * `delete/1` should return a `result` as String (1.2ms)
  * `get/0` should return a Map of tags to count (1.2ms)

PinboardixirTest


Finished in 0.2 seconds (0.1s on load, 0.1s on tests)
19 tests, 0 failures

Randomized with seed 476178
```

Getting Help

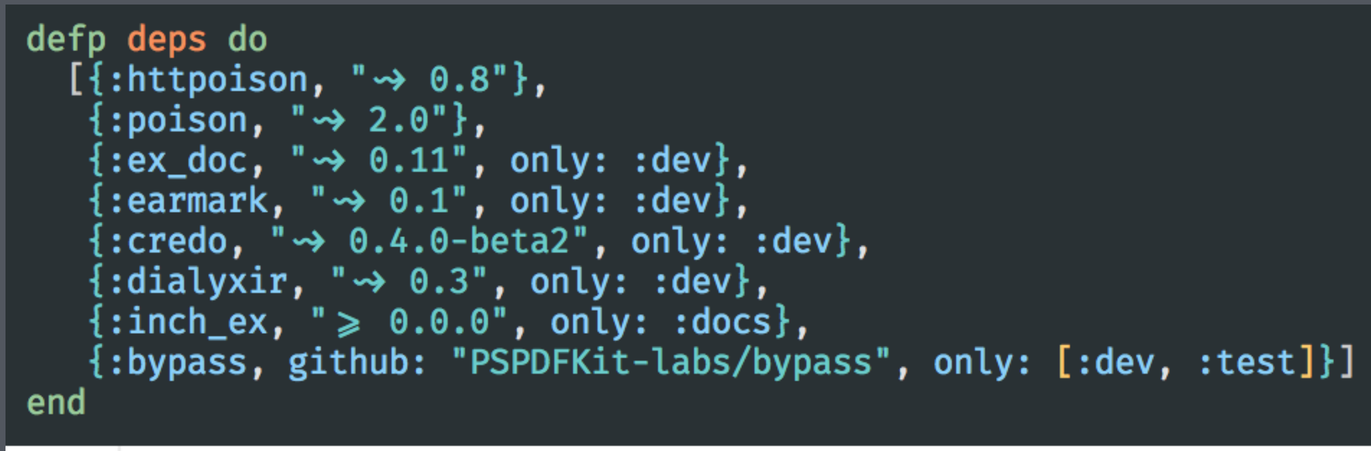
Finding Libraries

From:

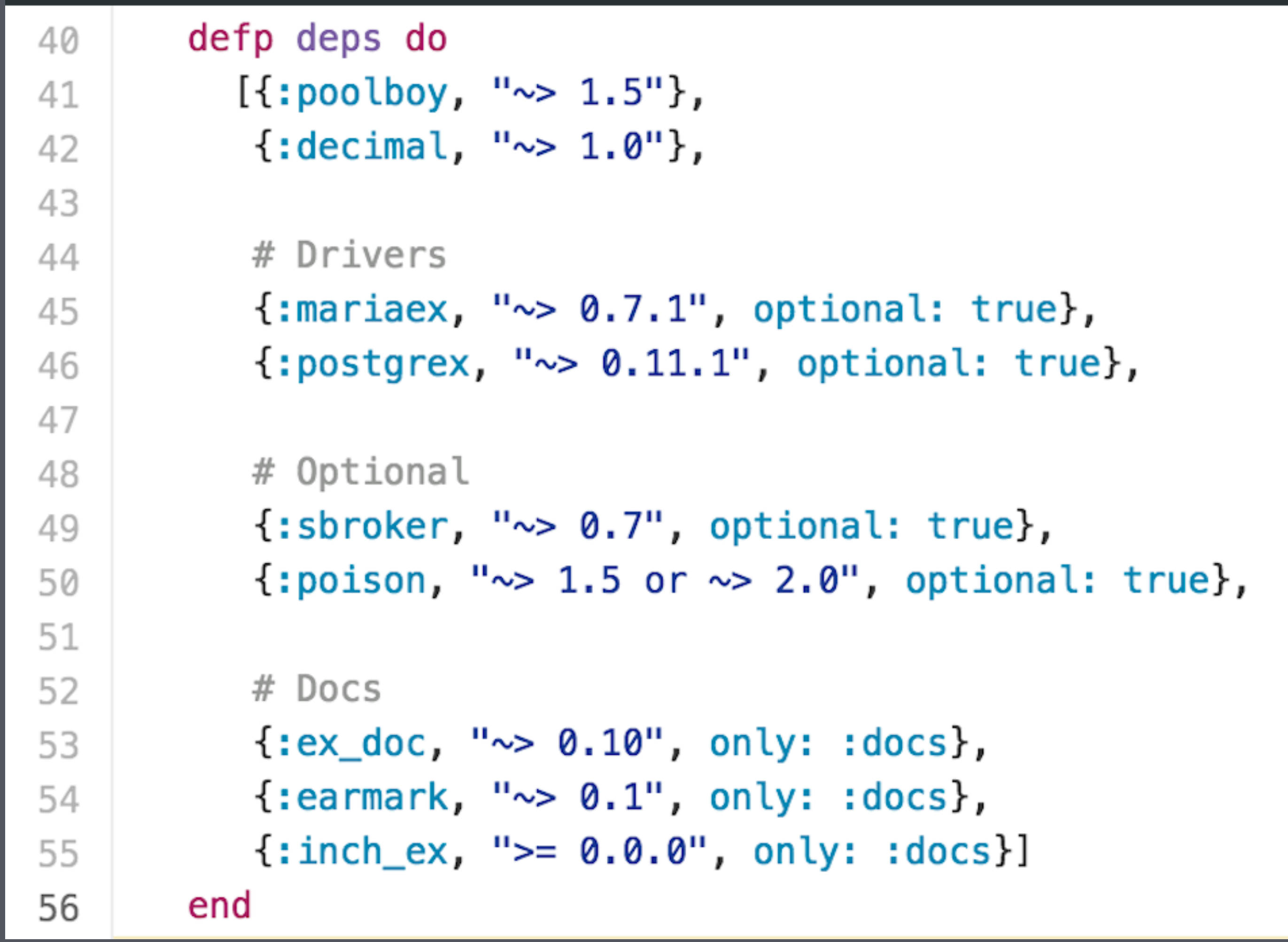
- awesome-elixir
- hex.pm, `mix hex.search`
- git, or GitHub
- rebar?
- Erlang libs

To:

- `:only`
- `:optional`



```elixir
defp deps do
  [{:httpoison, "~> 0.8"},
   {:poison, "~> 2.0"},
   {:ex_doc, "~> 0.11", only: :dev},
   {:earmark, "~> 0.1", only: :dev},
   {:credo, "~> 0.4.0-beta2", only: :dev},
   {:dialyxir, "~> 0.3", only: :dev},
   {:inch_ex, ">= 0.0.0", only: :docs},
   {:bypass, github: "PSPDFKit-labs/bypass", only: [:dev, :test]}]
end
```

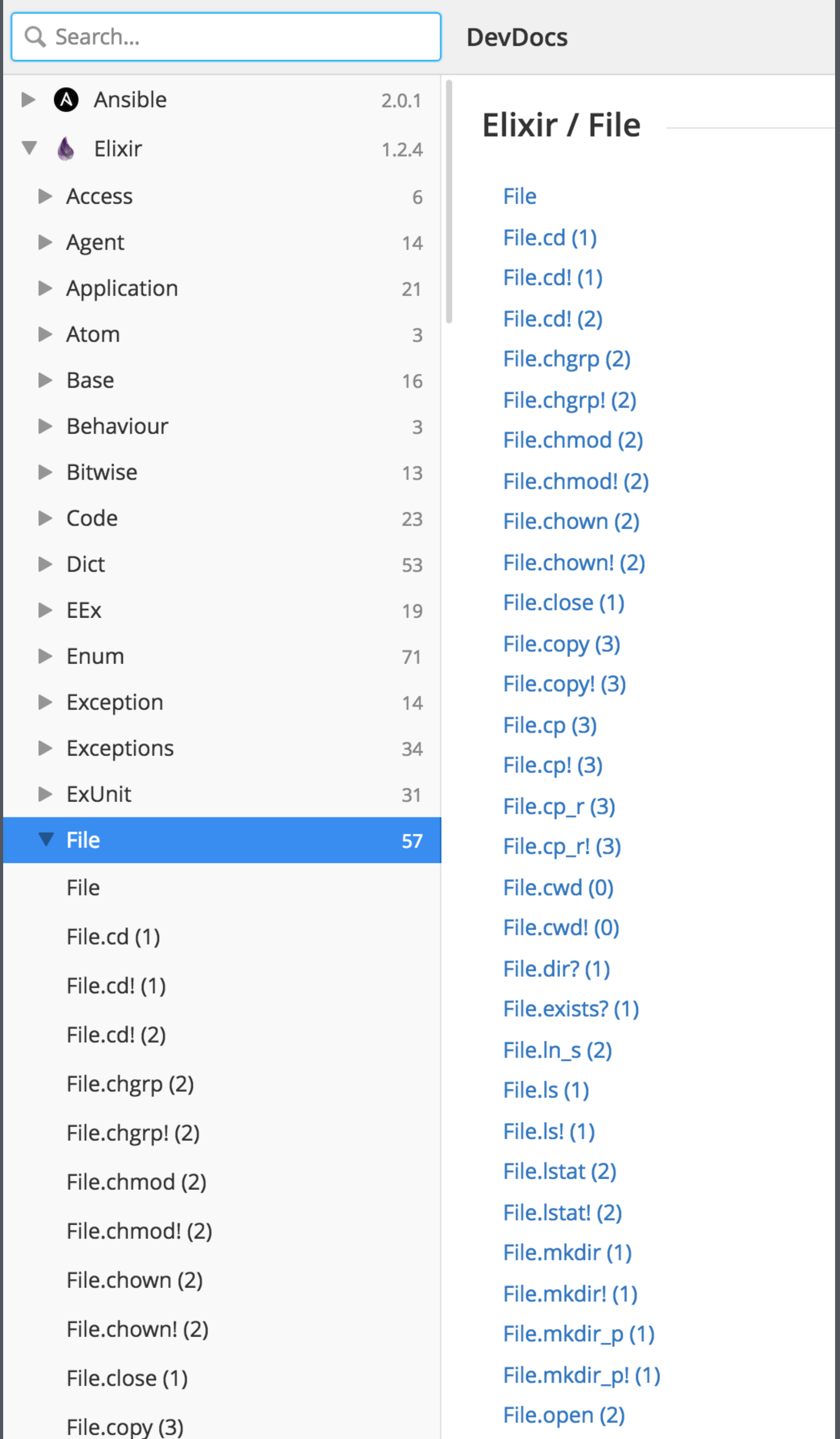


```elixir
defp deps do
  [{:poolboy, "~> 1.5"},
   {:decimal, "~> 1.0"},

   # Drivers
   {:mariaex, "~> 0.7.1", optional: true},
   {:postgrex, "~> 0.11.1", optional: true},

   # Optional
   {:sbroker, "~> 0.7", optional: true},
   {:poison, "~> 1.5 or ~> 2.0", optional: true},

   # Docs
   {:ex_doc, "~> 0.10", only: :docs},
   {:earmark, "~> 0.1", only: :docs},
   {:inch_ex, ">= 0.0.0", only: :docs}]
end
```

Documentation

- Elixir, EEx, IEx, Mix, ExUnit, Logger
- Plug, Ecto, ExDoc… and Phoenix
- devdocs.io
- hexdocs.pm
- h()

Lou Xun aquarhead@ela.build / Ela Workshop

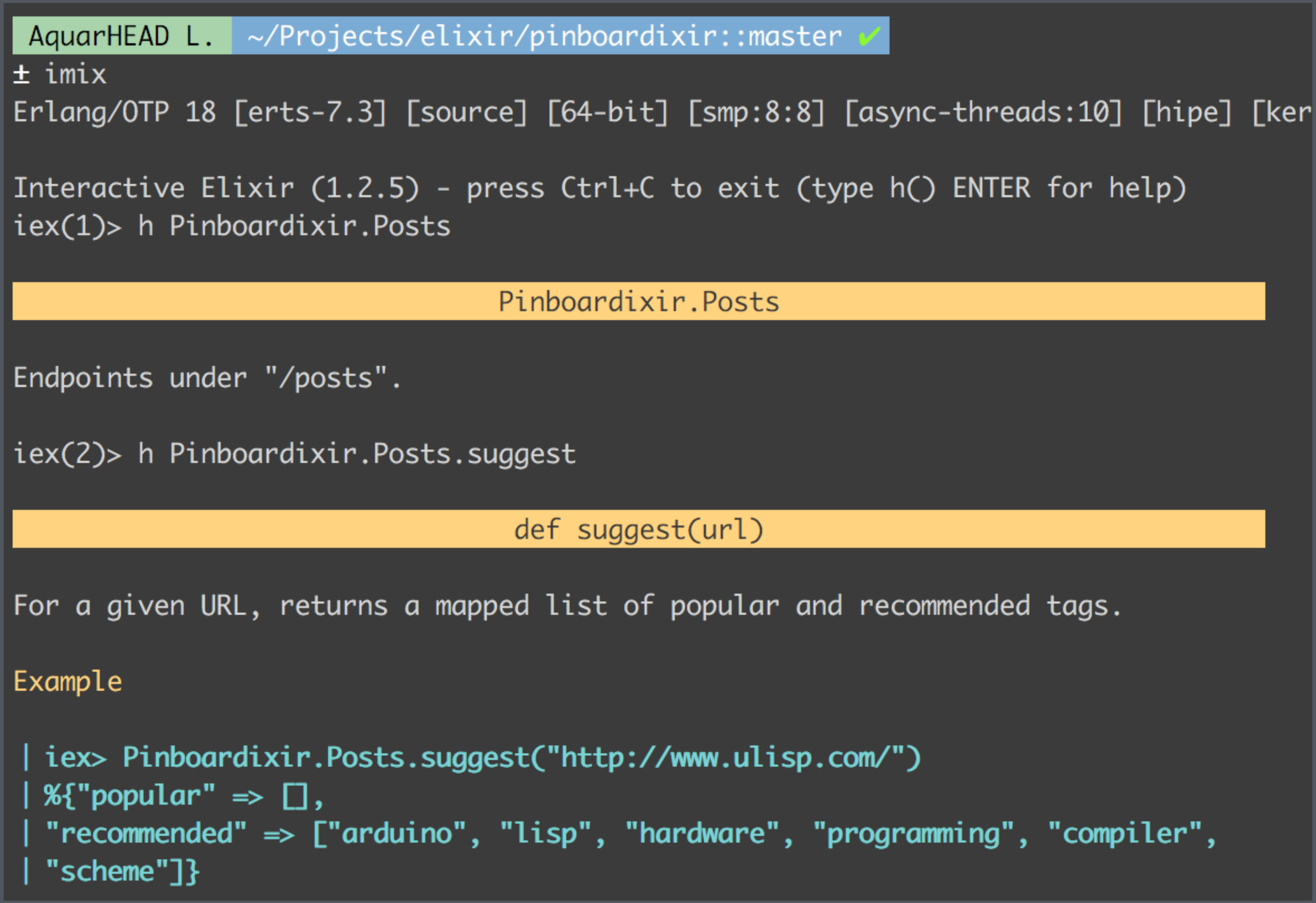


Ansible	2.0.1
Elixir	1.2.4
Access	6
Agent	14
Application	21
Atom	3
Base	16
Behaviour	3
Bitwise	13
Code	23
Dict	53
EEx	19
Enum	71
Exception	14
Exceptions	34
ExUnit	31
File	57
File	
File.cd (1)	
File.cd! (1)	
File.cd! (2)	
File.chgrp (2)	
File.chgrp! (2)	
File.chmod (2)	
File.chmod! (2)	
File.chown (2)	
File.chown! (2)	
File.close (1)	
File.copy (3)	

Elixir / File

File
File.cd (1)
File.cd! (1)
File.cd! (2)
File.chgrp (2)
File.chgrp! (2)
File.chmod (2)
File.chmod! (2)
File.chown (2)
File.chown! (2)
File.close (1)
File.copy (3)
File.copy! (3)
File.cp (3)
File.cp! (3)
File.cp_r (3)
File.cp_r! (3)
File.cwd (0)
File.cwd! (0)
File.dir? (1)
File.exists? (1)
File.ln_s (2)
File.ls (1)
File.ls! (1)
File.lstat (2)
File.lstat! (2)
File.mkdir (1)
File.mkdir! (1)
File.mkdir_p (1)
File.mkdir_p! (1)
File.open (2)

h()



```
AquarHEAD L.  ~/Projects/elixir/pinboardixir::master ✔
± imix
Erlang/OTP 18 [erts-7.3] [source] [64-bit] [smp:8:8] [async-threads:10] [hipe] [ker

Interactive Elixir (1.2.5) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> h Pinboardixir.Posts

                              Pinboardixir.Posts

Endpoints under "/posts".

iex(2)> h Pinboardixir.Posts.suggest

                               def suggest(url)

For a given URL, returns a mapped list of popular and recommended tags.

Example

│ iex> Pinboardixir.Posts.suggest("http://www.ulisp.com/")
│ %{"popular" => [],
│ "recommended" => ["arduino", "lisp", "hardware", "programming", "compiler",
│ "scheme"]}
```

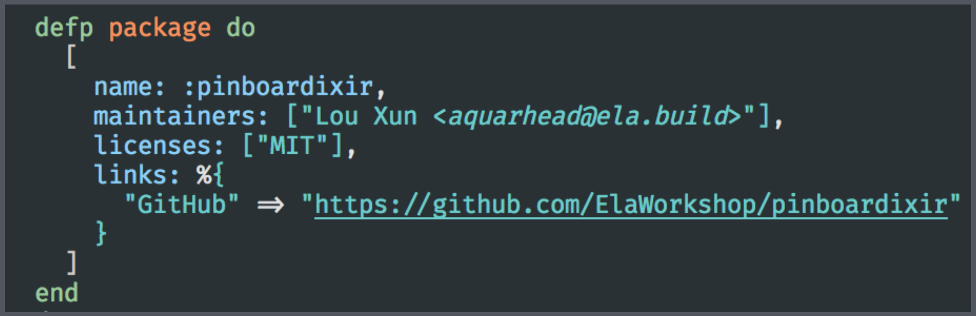
General Help

- Style Guide: `niftyn8/elixir_style_guide`
- Stack Overflow...
- IRC, Slack (#china), mailing lists
- Meetups
- Elixir Radar (weekly newsletter)

Contribute

Publish your Library

- hex.pm/docs/publish
- Register with `mix hex.user register`
- Add package metadata in `mix.exs`
- `mix hex.publish`
- `mix hex.docs`



```
defp package do
  [
    name: :pinboardixir,
    maintainers: ["Lou Xun <aquarhead@ela.build>"],
    licenses: ["MIT"],
    links: %{
      "GitHub" => "https://github.com/ElaWorkshop/pinboardixir"
    }
  ]
end
```

Contribute by Coding

- github.com/elixir-lang
- Other libs
- Other tools: devdocs.io, kiex, Dockerfile...

Contribute by Writing

- Elixir School
- Fix/Improve documentations
- Answer questions on Stack Overflow, etc..
- Your own blogs, etc..

Contribute by Talking

- Join Slack, IRC, mailing-list...
- Ask & answer
- Summarize good ones into gist, blog post..
- Give talks at Meetups!

Thanks