BOULDER ELIXIR MEETUP

PUBLISHING YOUR FIRST HEX PACKAGE

INTRODUCTION TO ELIXIR

TODD RESUDEK

Sr. Software Engineer at Weedmaps

supersimple

@sprsmpl

PUBLISHING YOUR FIRST HEX PACKAGE

WHAT IS A HEX PACKAGE?

Hex is the package manager for the Erlang ecosystem.

It is analogous to NPM for Node.js and RubyGems for Ruby.

WHAT IS A HEX PACKAGE?

Number of packages available:

- Hex: ~3,800
- RubyGems: ~8,300
- NPM: >250,000

INSTALLATION

If you have Elixir installed, chances are you already have hex installed.

You can check by typing:

```
mix hex -v
```

You should see an output like this:

```
Hex v0.15.0
Hex is a package manager for the Erlang ecosystem.
```

If not, you can install by typing:

```
mix local.hex
```

CREATING AN ACCOUNT

Before you can submit a package to hex.pm, you will need an account.

To begin the registration process, type:

```
mix hex.user register
```

Enter your email address and select a username and password.

Check your email to confirm your account.

CREATE AN ACCOUNT

WHAT'S IN A NAME?

Avoid using offensive or harassing package names, nicknames, or other identifiers that might detract from a friendly, safe, and welcoming environment for all.

CHOOSE A NAME

COMMON PRACTICES

If you are extending the functionality of another library, use that library name as a prefix.

For example, if you are adding XML parsing to `Poison`, consider a name like: `poison_xml`.

If you are porting a library from another ecosystem, it is common to prepend (or less commonly append) "_ex" to that library's name. For example, the AWS ruby gem is called "ex_aws" in hex. There is a Spotify package named "spotify_ex".

CHOOSE A NAME

COMMON PRACTICES

If you are porting a library from another ecosystem, it is common to prepend (or less commonly append) "`_ex`" to that library's name.

For example, the AWS ruby gem is called "`ex_aws`" in hex.

There is a Spotify package named "`spotify_ex`".

CHOOSE A NAME

CREATE AN EMPTY PROJECT

```
mix new <yourprojectname>
```

You should see output like this:

```
* creating README.md
* creating .gitignore
* creating mix.exs
* creating config
* creating config/config.exs
* creating lib
* creating lib/yourprojectname.ex
* creating test
* creating test/test_helper.exs
* creating test/yourprojectname_test.exs
```

START CODING

DEFINE YOUR PACKAGE

Metadata will help define your package dependencies as well as describe it to potential users.

Open your mix.exs file and you'll see a template like this:

```
def project do
  [app: :yourprojectname,
   version: "0.1.0",
   elixir: "~> 1.4",
   build_embedded: Mix.env == :prod,
   start_permanent: Mix.env == :prod,
   deps: deps()]
end
```

ADDING METADATA

ADD METADATA

Hex packages are required to follow semantic versioning.

DEPENDENCIES

Dependencies are defined in a private method, `deps/0`.

This is where you will document any other packages your package requires (if any.)

For example, you might want to require `ex_doc` to generate documentation:

```
defp deps do
  [{:ex_doc, ">= 0.0.0", only: :dev}]
end
```

ADDING METADATA

DEPENDENCIES

After you add dependencies, you need to type: `mix deps.get` to include them in your project.

This is just like adding a gem to your gemfile and typing: `bundle install`.

To learn more about dependencies, type: `mix help deps`.

ADDING METADATA

DESCRIBE YOUR PACKAGE

In addition to the attributes predefined for you in your mix file, you will need to include a description and package information.

```
def project do
  [app: :yourprojectname,
   version: "0.1.0",
   elixir: "~> 1.4",
   build_embedded: Mix.env == :prod,
   start_permanent: Mix.env == :prod,
   description: description(),
   package: package(),
   deps: deps()]
end
```

START CODING

DESCRIBE YOUR PACKAGE

```
defp description do
  """
  Type your multi-line description here. This will be
  what users see when they view your package on hex.pm
  """
end
```

START CODING

DESCRIBE YOUR PACKAGE

```
defp package do
  [# These are the default files included in the package
   name: :yourprojectname,
   files: ["lib", "mix.exs", "README*", "LICENSE*"],
   maintainers: ["Your name and/or email"],
   licenses: ["GPL 3.0"],
   links: %{"GitHub" => "https://github.com/user/yourprojectname"}
   ]
end
```

START CODING

ADD A LICENSE

The full text of the license you choose should be in a file named `LICENSE.md` and placed at the root of your project.

You can find the descriptions and full text of open source licenses on https://opensource.org/licenses

LICENSE YOUR SOFTWARE

LET'S LOOK AT SOME CODE...

LIVE DEMO

COMPILE

Type: `mix compile`

This will compile and .ex files and generate documentation for your project.

COMPILE YOUR PROJECT

SUBMIT TO HEX.PM

Type: `mix hex.publish` to begin the submission process.

SUBMIT YOUR PACKAGE

DESCRIBE YOUR PACKAGE

You'll see some output as it reads your package.

Review it carefully.

SUBMIT YOUR PACKAGE

APPROVED

SOURCE CODE

github.com/supersimple/alphabetify-ex