JJ

Everyday Elixir

Joel Byler
@joelbyler

JJ

How I Learned Elixir

Joel Byler
@joelbyler

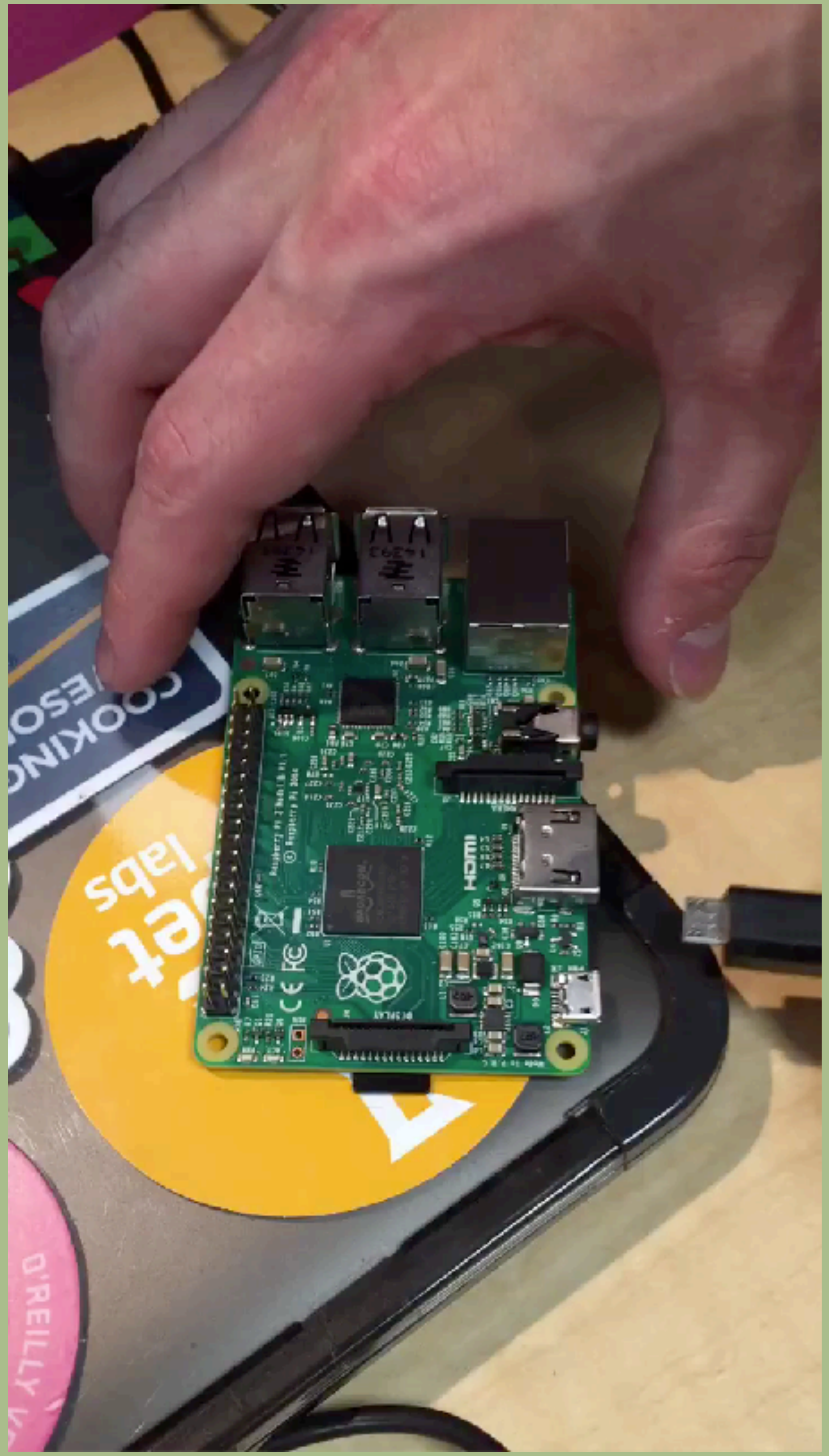
JJ

How I Learned Elixir …and I’m Still Learning

Joel Byler
@joelbyler

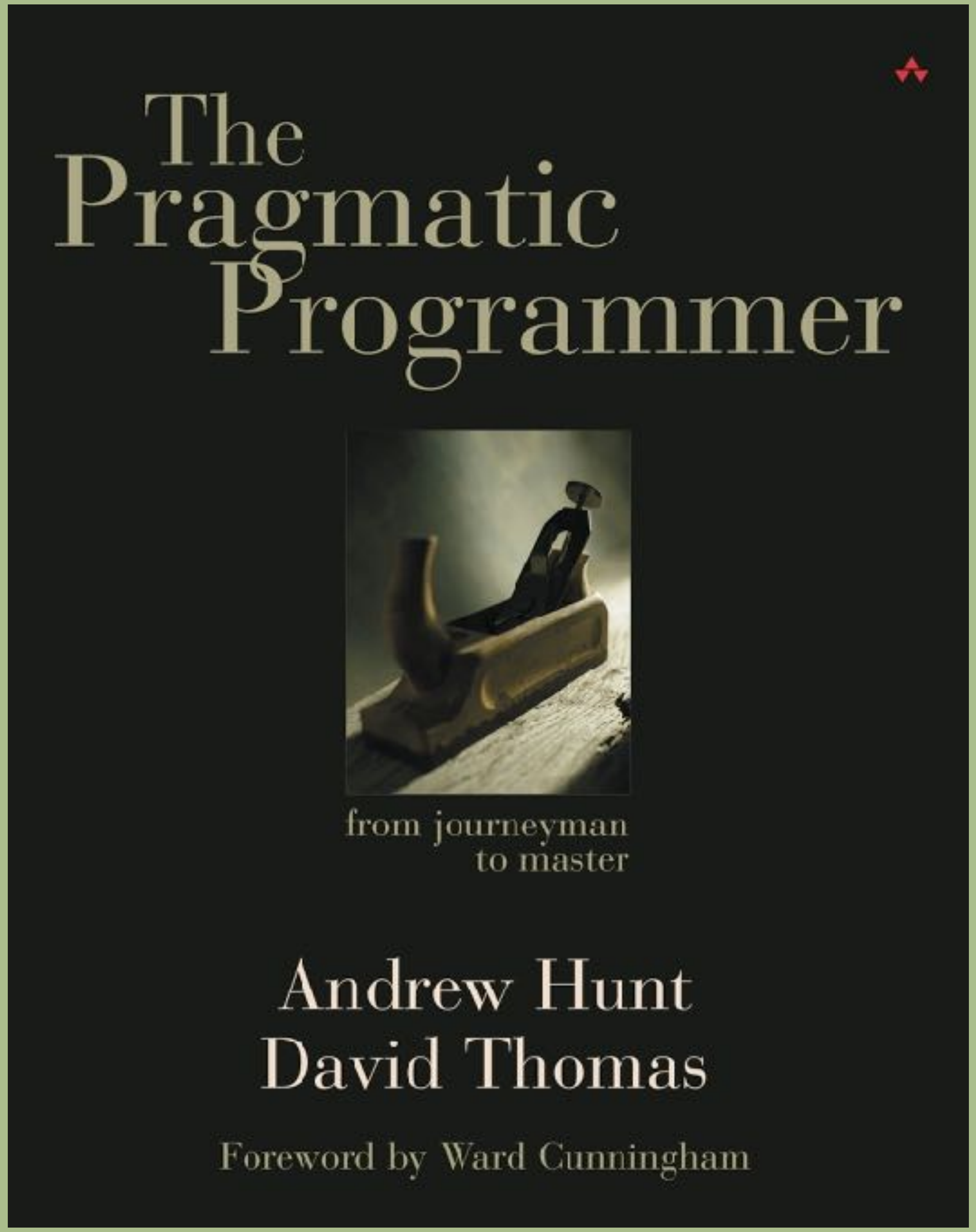
I work at CoverMyMeds!

- Helping patients get the medications they need live healthy lives
- Consistently rated best place to work in Central Ohio
- Mostly Ruby / Rails but have a few Elixir / Phoenix apps in prod
- Columbus, Cleveland, and Remote



https://twitter.com/Kumichou/status/692870886681026566

Why learn a new programming language?

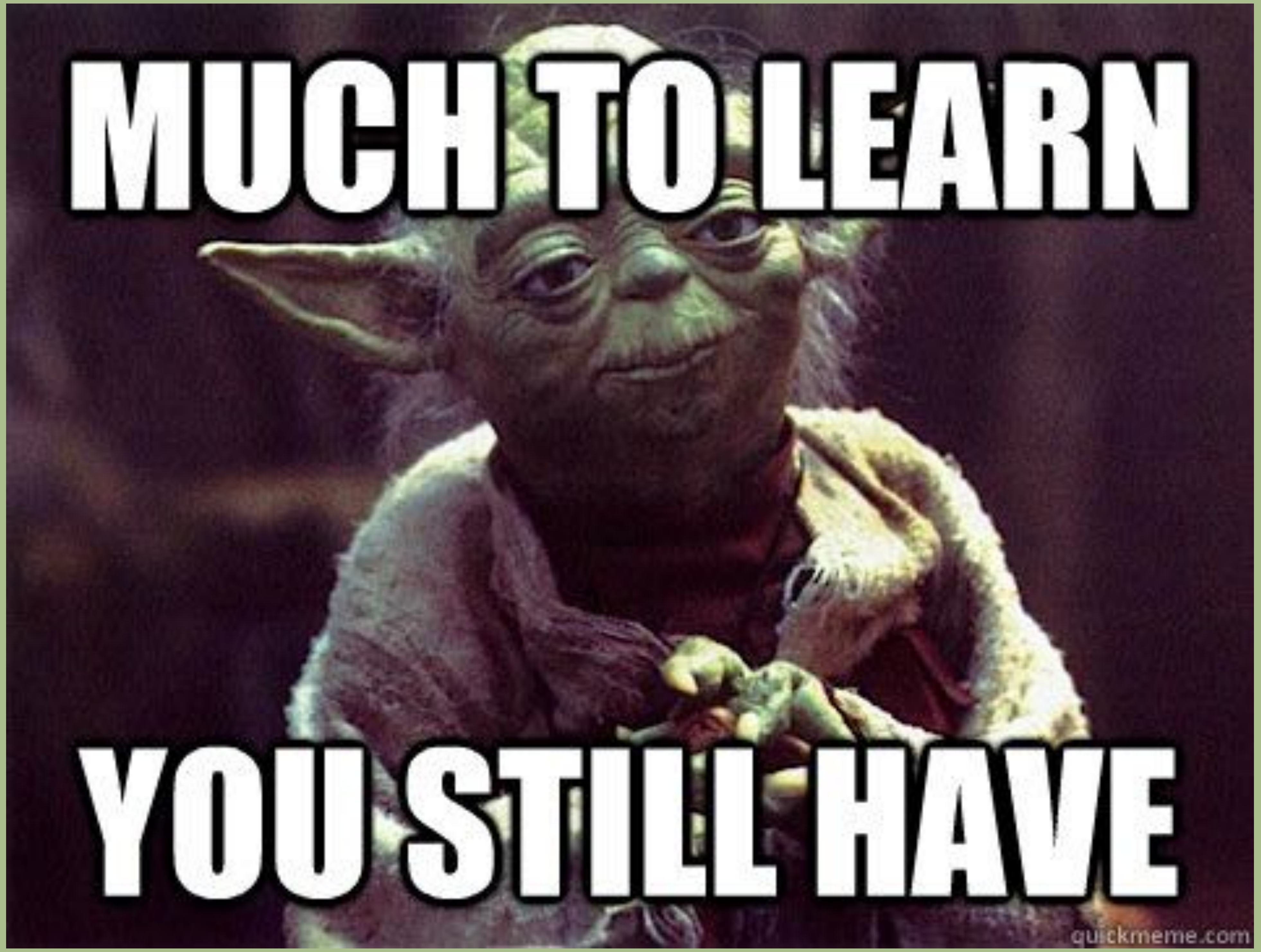


Invest Regularly in Your Knowledge Portfolio

Make learning a habit.

https://imgflip.com/i/1zy44j

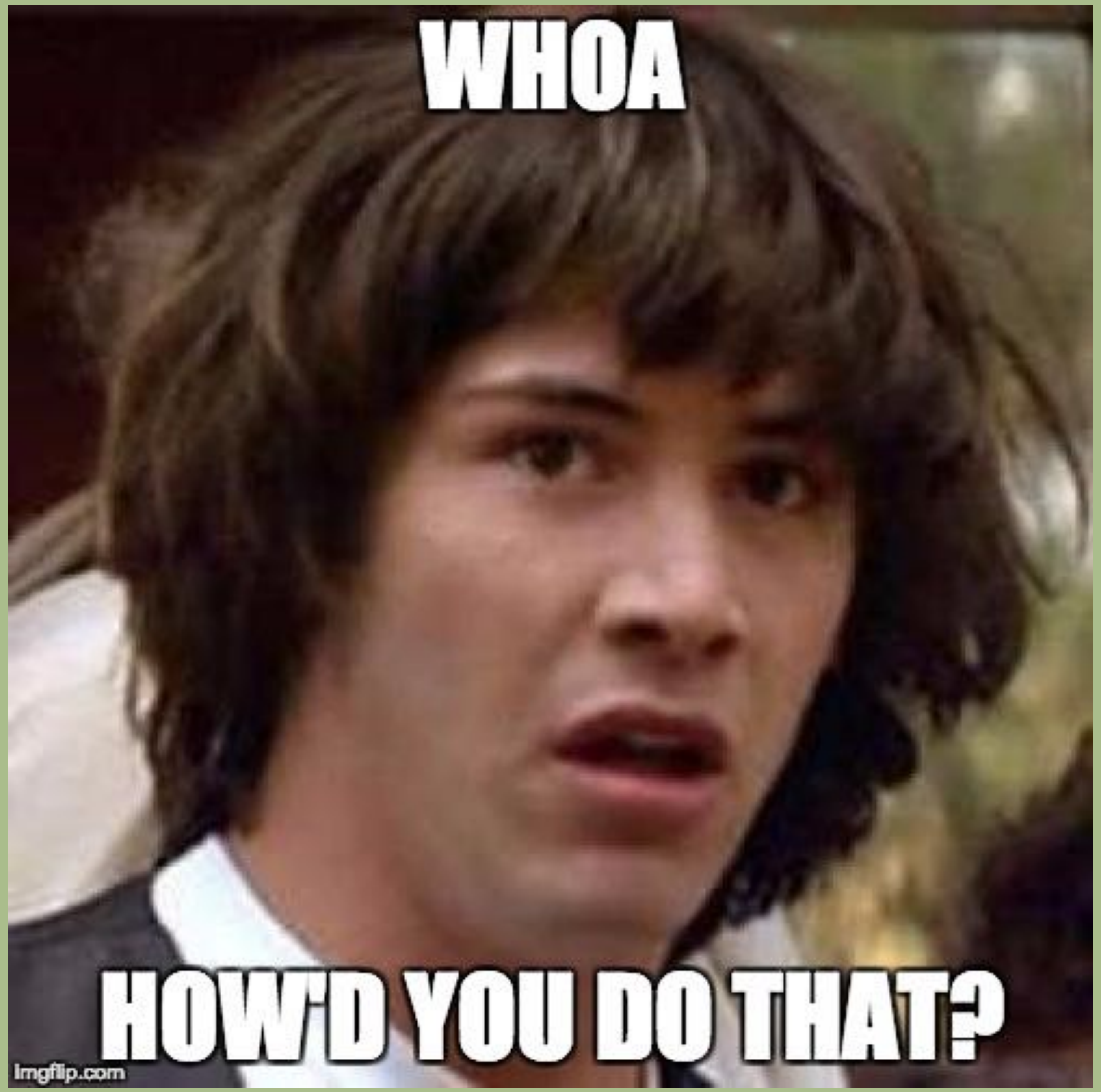
Why learn a new programming language?



Practice / improve learning skills

https://i.pinimg.com/originals/46/88/3a/46883aa8be123375b2e45a3d2bd2ec23.jpg

Why learn a new programming language?



Improve understanding of the benefits of using a different programming language

https://imgflip.com/i/1zy44j

Why learn a new programming language?

FRIENDSHIP IS SO WEIRD, YOU JUST PICK A HUMAN YOU'VE MET AND YOU'RE LIKE
«YUP I LIKE THIS ONE» AND YOU JUST DO STUFF WITH THEM.

Make new friends!

https://i.pinimg.com/736x/d5/23/85/d5238507a99460d33528ea1fdf0c8946--funny-but-true-lol-funny.jpg

Why learn a new programming language?

look at problems from a different perspective

https://pbs.twimg.com/media/B2S3V0sIcAAbnNJ.png

Why learn a new programming language?

Possibly open up the door to new opportunities?

https://www.landispr.com/blog2/wp-content/new-job-meme.jpg

Types of Learners

Neil Fleming's VARK model

- **Visual** - visualize relationships and ideas
- **Auditory** - prefer listening to information rather than reading or seeing it visually displayed
- **Reading / writing** - extremely comfortable with written word, books, blogs and blogging
- **Kinesthetic** - hands-on learning, big fans of the CodeMash precompilers

http://vark-learn.com/

Types of Learners

Neil Fleming's VARK model

- **Visual** - visualize relationships and ideas
- **Auditory** - prefer listening to information rather than reading or seeing it visually displayed
- **Reading / writing** - extremely comfortable with written word, books, blogs and blogging
- **Kinesthetic** - hands-on learning, big fans of the CodeMash precompilers

http://vark-learn.com/

So what about Elixir?

pipe |>

pipe |>

WITHOUT PIPES:

```
function_3(function_2(function_1("params")))
```

pipe |>

WITHOUT PIPES:

```
function_3(function_2(function_1(“params”)))

OR

function_3(
 function_2(
  function_1(“params")
 )
)
```

pipe |>

WITHOUT PIPES:

```
function_3(function_2(function_1(“params”)))

OR

var1 = function_1(“params”)
var2 = function_2(var1)
var3 = function_3(var2)
var3
```

pipe |>

WITH PIPES:

```
function_1("params")
 |> function_2
 |> function_3
```

pipe |>

WITH PIPES:

```
function_1("params")
 |> function_2
 |> function_3

OR

"params"
 |> function_1
 |> function_2
 |> function_3
```

pattern matching

{:ok, “so cool”}

pattern matching

WITHOUT PATTERN MATCHING:

```
def some_function(param) do
  if param == 1 do
    "yay"
  else
    "nay"
  end
end
```

pattern matching

WITH PATTERN MATCHING:

```
def some_function(1) do
   "yay"
end
def some_function(_) do
   "nay"
end
```

pattern matching

WITH PATTERN MATCHING:

```
def some_function(1) do
   "yay"
end
def some_function(_) do
   "nay"
end

OR

def some_function(1), do: "yay"
def some_function(_), do: "nay"
```

pattern matching

WITH PATTERN MATCHING:

```
case parse_messages(params) do
  {:ok, %{messages: messages}} ->
   {:ok, messages}

  {:ok, _} ->
   {:ok, "No messages"}

  {:error, :invalid_format} ->
   {:error, "Invalid format"}

  {:error, _} ->
   {:error, "Invalid inputs"}
end
```

pattern matching

WITH PATTERN MATCHING:

```
with
  {:ok, body} <- build_params(params),
  {:ok, headers} <- headers(auth_params),
  result = client.post(url, body, headers),
  {:ok, %{body: resp_body,
          status_code: 200}} <- result,
  {:ok, json_data} <- decode(resp_body)
  do:
   {:ok, json_data}
else
   {:error, %{}}
end
```

application config

general / dev / test /prod

configuration

config/config.exs

```
config :camera,
  adapter: Picam,
  image_path: "pic_images/",
  image_location: "/root/images"

import_config "#{Mix.env}.exs"
```

config/dev.exs

```
config :camera,
  adapter: Fake.Picam,
  image_location: "static/sample_images"
```

IEx
for the kinesthetic learner

IEx

```
$ iex
iex(1)> bar = "world"
"world"
iex(2)> "hello #{bar}"
"hello world"
iex(3)> 1 + 1
2
iex(4)>
```

IEx

```
defmodule Demo do
  def foo(bar) do
    require IEx; IEx.pry
    "hello #{bar}"
  end
end
```

```
$ iex -S mix phx.server
iex(1)> bar
"hello world”
iex(2)> 1 + 1
2
iex(3)> respawn()
```

Practical Applications

Problem: Checking status pages on a regular basis

Solution: Elixir script to automate opening the tabs in Chrome

Auto Browser Tabs
(very simple)

```
System.cmd("open", [
  "https://status.github.com/messages",
  "https://coinbase.statuspage.io/",
  "http://www.vimeostatus.com/",
  "https://status.shopify.com/",
  "http://status.digitalocean.com/",
  ])
```

Learned: simple task automation using scripts

Problem: I want to automate the task of checking those status pages

Solution: Write a test to assert against content of status pages

Status Page Checks
(automated browser checks)

```
test "check pusher status" do
  navigate_to("https://status.pusher.com/")

  status =
   find_element(:class, "page-status")
    |> visible_text
    |> String.trim

   assert(status == "Systems Operational")
end
```

Learned: how to build a feature spec using ex_unit and hound

Problem: I want to check the CodeMash schedule quickly

Solution: write a CLI to query the CodeMash schedule for a topic

Command Line (CLI)
(codemash cli)

```
defmodule CodemashCli.CLI do
  def main(args) do
    parse_args(args)|> process
  end
  def process(:help) do
    Mix.shell.info """
    Codemash CLI
    - - - - -
    usage: codemash_cli <search_term>
    example: codemash_cli elixir
    """
  end
  def process(search_term) do
    CodemashCli.Query.fetch(search_term)
    |> CodemashCli.ExtractMap.extract_from_body
  end
end
```

Learned: how to use escript to build a CLI

Problem: Our current attendance tracking software is too $$$

Solution: Build a custom web app to track attendance

MVC Web App (phoenix)

```
defmodule LifehopeAttendance.EventController do
  use LifehopeAttendance.Web, :controller

  alias LifehopeAttendance.Event

  def index(conn, _params) do
    events = Repo.all(Event)
    render(conn, "index.html", events: events)
  end

  def new(conn, _params) do
    changeset = Event.changeset(%Event{})
    render(conn, "new.html", changeset: changeset)
  end
end
```

Learned: how to build a simple phoenix app and deploy it to heroku

Problem: We need a way to estimate our work collaboratively

Solution: Build another web app with collaborative features

MVC Web App
(phoenix + channels)

```
defmodule PointingParty.PartyChannel do
  use Phoenix.Channel
  alias PointingParty.Presence

  def join("party:" <> party_key, _params, skt) do
    party = PointingParty.PartyTracker.party(party_key)
    ...
    send self(), :after_join

    {:ok, %{user_name: socket.assigns.user, data: %{}}, skt}
  end

  def handle_info(:after_join, skt) do
    push socket, "presence_state", Presence.list(skt)
    ...
    {:noreply, skt}
  end
end
```

Learned: how to build a phoenix 1.3 app using channels

Problem: I want to document my CodeMash experience

Solution: Build a time-laps camera!

Embedded Linux (nerves)

```
def deps(target) do
  [
    {:bootloader, "~> 0.1"},
    {:nerves_runtime, "~> 0.4"},
  ] ++ system(target)
end

def system("rpi"), do:[{:nerves_system_rpi, ">= 0.0.0"}]
def system("rpi0"),do:[{:nerves_system_rpi0,">= 0.0.0"}]
def system("rpi3"),do:[{:nerves_system_rpi3,">= 0.0.0"}]
def system("bbb"), do:[{:nerves_system_bbb, ">= 0.0.0"}]
```

Learned: how to use nerves to build a wearable time laps camera

Who’s up for a #fridayhug ?

Questions?

Resources

- ElixirConf YouTube Channel: https://goo.gl/C9jQGQ
- The Little Elixir & OTP Guidebook
  by Benjamin Tan Wei Hao
- Programming Phoenix: Productive |> Reliable |> Fast
  by by Chris McCord, Bruce Tate, Jose Valim
- ElixirWeekly Mailing List: https://elixirweekly.net/
- Elixir Slack Community
  https://elixir-slackin.herokuapp.com/
- Twitter - #myelixirstatus

THANK YOU!

All source code available from

https://github.com/joelbyler/codemash_2018

Joel Byler
@joelbyler

SDG