Started From The Bottom, Now We're Here

(Or: Self-Improvement via OSS)

github.com/mtrudel/talks

Why are we here?

- Lots of new folks asking how to get a foot in the door / level up their career
- A **huge** part of career advancement is personal development
- OSS is a **perfect** avenue for this
- THIS IS JUST ONE (SUPER PRIVILEGED) GUY'S OPINION
- THIS IS JUST ONE POSSIBLE PATH

What career growth (often) looks like

Started From The Bottom

- Your goal is **'any job that pays'**
- Employer holds all the cards
- Hiring evidence is **skills-based**
- Ancillary skills are less important

...Now We're Here

- Your goal is the **job you want**
- Relationship becomes **more mutual**
- Hiring evidence is **portfolio-based**
- Ancillary skills are **all** that matter

Experience / Seniority / Time

WTF
does this have to do with
OSS?

A huge part of
career advancement
is
personal development

OSS is a superb way to facilitate this

'Any job that pays' -> The job you want

- What **kind** of job do you want?
  - The **overhead** of changing jobs is **large**
  - OSS projects are **free**
- **Try new things** on with close to zero commitment
- Natural avenue to **grow into an authority** in your domain
- Build your **future network** along the way

Towards a more mutual work relationship

- As your value to employers grows, so does your **bargaining position**
- This becomes your **primary leverage** for promotions, raises, new jobs
- BUT, this requires:
  - Having the **context** to know what you're worth and what other opportunities exist
  - Having the **communication** and **people skills** to negotiate for yourself
  - Having realistic and obvious **alternative options** (BATNA)

Towards a more mutual work relationship

- OSS helps with this in several ways:
  - Provides an avenue for you to **develop skills** that are directly applicable in your current job
  - Allows you to build a **network** and **personal brand** that helps you find a new job
  - When you are seen to build value outside of work, work gets FOMO

Building Your Portfolio

- Goal is to have your portfolio serve as evidence of your competency
- GitHub is likelier than not the **core** of your portfolio
- But your portfolio is a **lot** more than just your code!
  - How do you write & communicate?
  - What sort of community member are you?
  - Ancillary technical skills are also super important (OSS is great for this!)

Soft Skills > Hard Skills

- Being able to communicate / persuade / inspire / charm is a massive multiplier
- Effective examples vary with the medium
  - Communicating clearly on a GitHub issue
  - Saying 'no' (or accepting one!) without discouragement
  - Negotiating a compromise on a feature / line of responsibility at work
- Learning to do this genuinely is one of the best life skills you can cultivate

In Review

- Hard skills get you your first job, but soft skills move you up the ranks
- Blending them effectively is an art not a science
- Being genuine and true in everything you do is an invariant
- Building your soft skills while honouring this is basically the key to life
- Why **not** practice it in public?

Working in public can take **many** different forms

Working in public can take **many** different forms

- Starting a new project as a founder
  - Are you here for a good time or a long time?
- Contributing to existing projects
  - High profile work? Thankless maintenance?
- Developing educational content
  - Streams, podcasts, blogs, conference talks
- Maybe a mix of all three?

What is it that you want to accomplish?

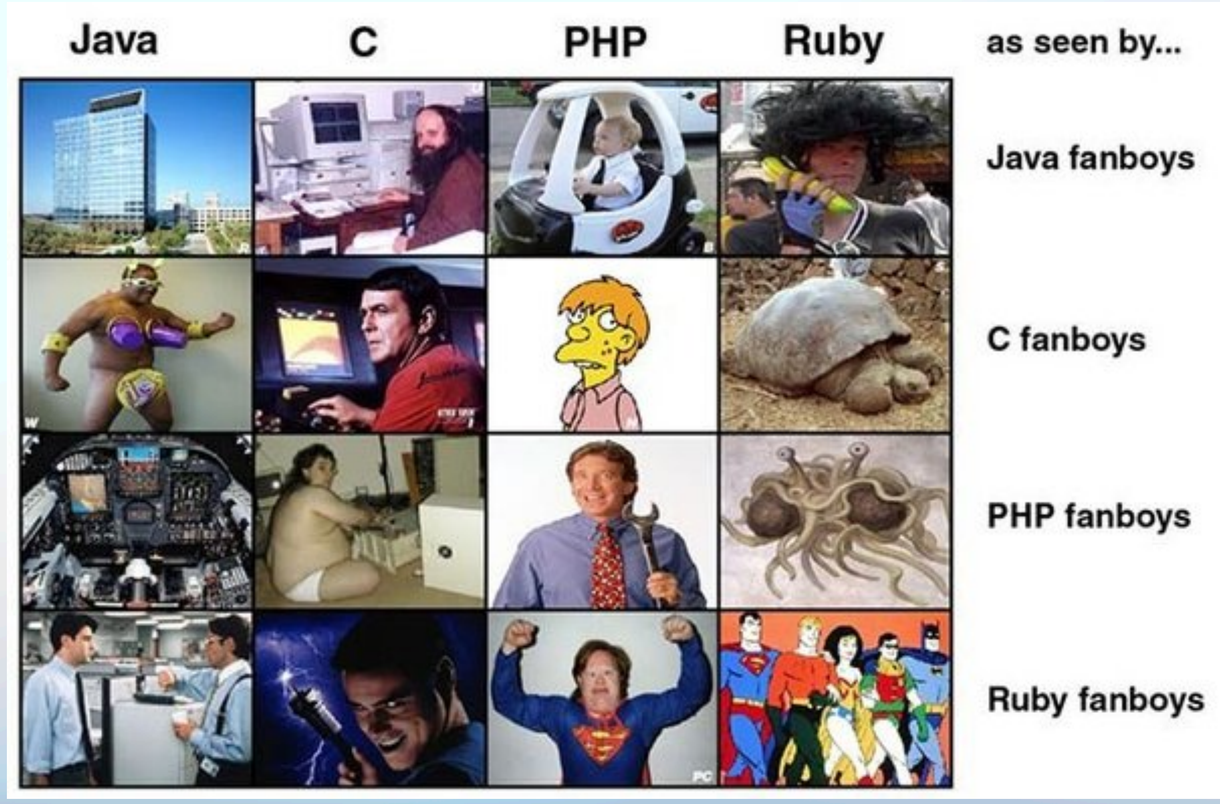
- Growing your skills & rep in your primary language / community
- Learning a new primary language / framework
- Building just for the sake of it
- All of these have tradeoffs
  - Immediate relevance vs. longer term growth
  - Where are you building your network & rep?

How are **you** shaped by your **ecosystem**?

Ecosystems have a shape

Ecosystems have an **ethos**

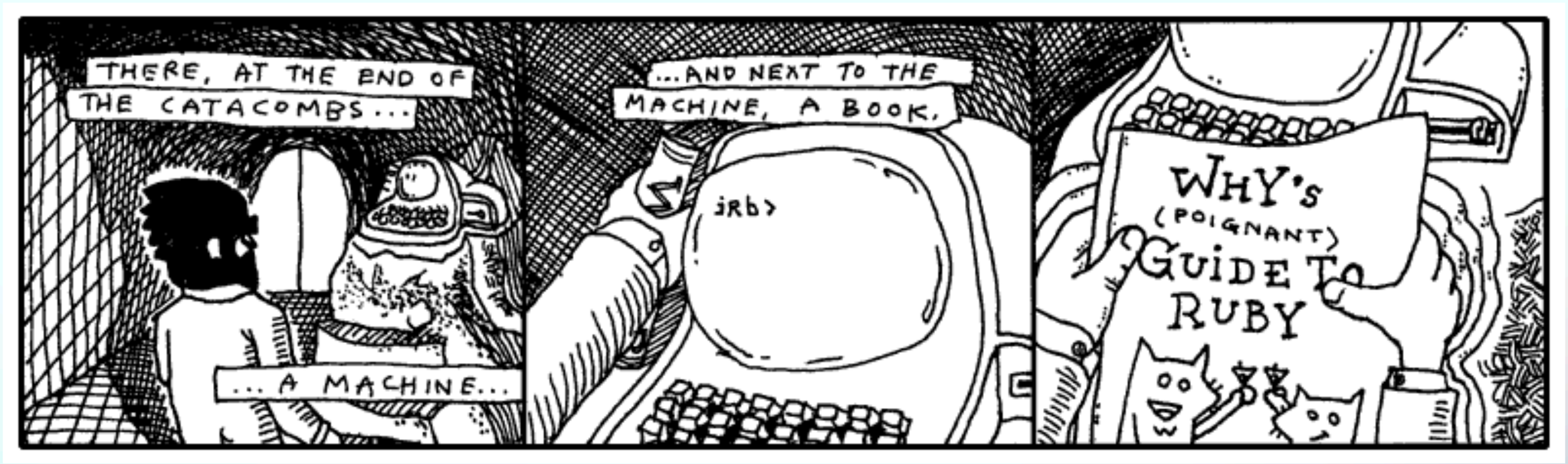
- Javascript: attention span of a goldfish, in love with complexity, technology's English
- Python: accessible, opinionated, ubiquitous
- Java: top-down, high ceremony, boring (in a good way)
- Ruby: joyful, pragmatic, a cargo-cult
- Swift: whatever Apple wants it to be
- The LISPs: elegant, correct, obscure
- Elixir: joyful, pragmatic, unconventional



Your ecosystem will influence...

- The scope of your work
- Your sense of what's possible
- Your sense of agency
- Your sense of urgency
- Your perspective on authority
- How you work with others

Elixir owes a
huge
cultural debt to Ruby



poignant.guide

MINSWAN

(Matz Is Nice So We Are Nice)

Working In Public

Nadia Eghbal

"when you don't create things, you become defined by your tastes rather than ability. your tastes only narrow & exclude people. so create."

- why the lucky stiff

Working In Public

(Or: One way to effectively practice OSS)

github.com/mtrudel/talks

My experience with Bandit

Bandit is an HTTP server for Plug Applications

github.com/mtrudel/bandit

- Net-new project, founded for the long haul. Mid-profile
- Mostly grinding on development initially, gradually shifting to maintenance
- Working in my primary language, though the technical problems are new
- Deeply embedded in the broader ecosystem (e.g. Phoenix, Elixir core, HTTP community)
- Lots of educational / PR work (half a dozen talks, countless podcasts, forum activity, etc)
- Roughly estimate 1200-1500 hours of personal effort over four years

Let's take a look

Starting
Growing
Maintaining

Starting

Your biggest problem will be scoping & motivation

What is your charter?

- What does **"done"** look like? *(full RFC support, obvious best option for hosting Phoenix)*
- Published roadmap? *(yes, but mostly for myself)*
- What are your high-level goals & principles *("correctness, clarity, performance, in that order")*
- Do you have any red lines? *(OTP as gospel, minimum of codified policy)*
- Stance on contributions & governance *(yes please! BDFL-lite maintainer role)*
- Stance on community obligation *(extremely high; Bandit is foundational)*

RDD

Write your README first

Start high-level, work down

1. Build the simplest end-to-end skeleton you can
2. Identify the piece needing the most attention
3. Make that piece 'good enough'
4. Repeat

Keep doing this until you have something **useful**

(0.1.0 to 0.x.y)

Bandit took about a year to 'start'

- First commit Nov 2019
- Initial version was the simplest thing that worked
- Became minimally useful in about a week, BUT
- Stopped working on it for months at a time (working on HAP)
- Burned it down and rebuilt it multiple times
- 0.1.0 released Nov 2020

In the beginning, nobody cares

(this is freeing)

Growing

Begins when your project is

minimally useful to someone

A new set of concerns

- Development: you're still building greenfield at this stage
  - BUT: now people have opinions
- PR & adoption: getting people to know / care about your project is HARD
- Support: people make some crazy requests
- It's up to you to determine what all of this means to you
  - BUT: the world doesn't care about your feelings, only what you can do for them

Start tightening up your work

- A good 'quickstart' section in the README
- Docs for at least the primary parts of your library
- No more *'works on my machine'*
- At least basic CI
  - github.com/mtrudel/elixir-ci-actions makes this trivial

Start talking about your work

"Do Things, Tell People"

- Post on the Elixir Forum
- Give a meetup or conference talk (it's a lot easier than you think!)
- Get Tyler to write a blog post about your library
- Be polite, gracious and patient with issues and PRs
- This is a prime place for developing those soft skills

Start integrating your work

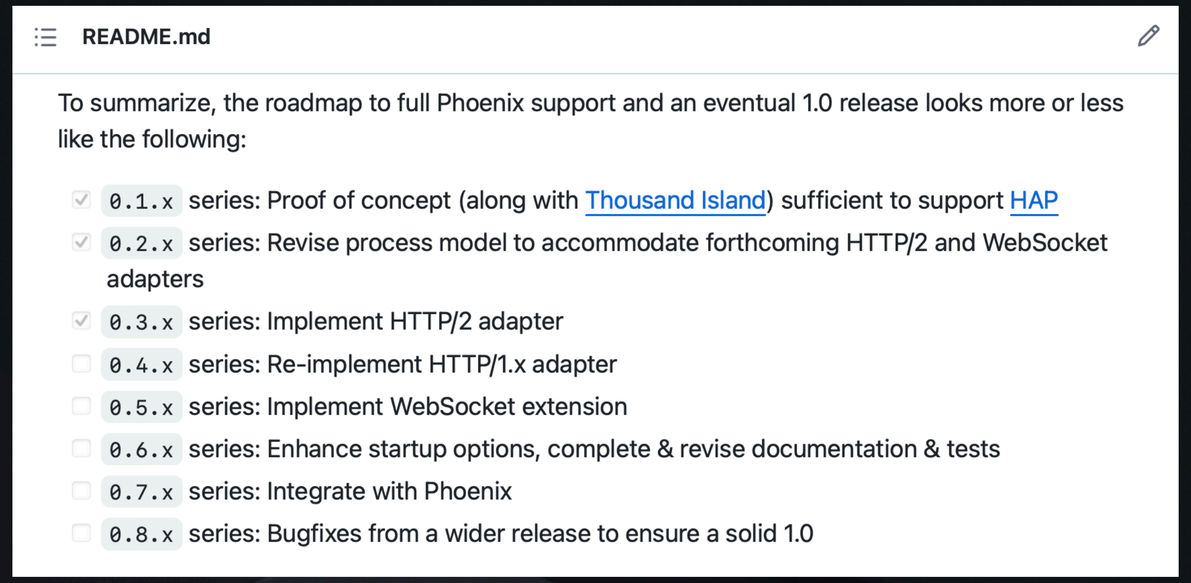
- Collaborate up and down the stack. Get to know your neighbours!
  - Interact with their PRs!
  - File fixes!
  - Do their integration work for them!
  - Get people to know you exist!
  - Have a personality! Stupid hacks like a distinctive avatar or standard signoff help

Keep doing this until you have something **complete**

(0.x.y to just before 1.0.0)

Bandit 'grew' for about 2½ years

- 0.1.0 in Nov 2020
- Public 'unveiling' at ElixirConf in Oct 2021
- First external PR Oct 2021 (Thanks Travis!)
- HTTP/2 support added in May 2022
- Phoenix integration via WebSock in Nov 2022
- 1.0.0-pre release series started Apr 2023


README.md

To summarize, the roadmap to full Phoenix support and an eventual 1.0 release looks more or less like the following:

- [x] `0.1.x` series: Proof of concept (along with Thousand Island) sufficient to support HAP
- [x] `0.2.x` series: Revise process model to accommodate forthcoming HTTP/2 and WebSocket adapters
- [x] `0.3.x` series: Implement HTTP/2 adapter
- [ ] `0.4.x` series: Re-implement HTTP/1.x adapter
- [ ] `0.5.x` series: Implement WebSocket extension
- [ ] `0.6.x` series: Enhance startup options, complete & revise documentation & tests
- [ ] `0.7.x` series: Integrate with Phoenix
- [ ] `0.8.x` series: Bugfixes from a wider release to ensure a solid 1.0

Maintaining

Begins when your project has

'nothing' left to do

Usually corresponds to

1.0.0

Bandit has been in 'maintenance' for 6 months

- Started 1.0.0-pre in April 2023
- Released 1.0.0 in Oct 2023
- Adoption & awareness skyrocketed after 1.0.0
- Integrations coming fast
- Support burden is very repetitive
- I started the 1.0.0-pre series too early (mostly a process problem with SemVer)

A new set of concerns

- Development: SemVer mandates being careful with your changes
- Support: you'll start seeing the same issues come up over and over
- More eyeballs: you'll start getting more and better suggestions
- Contributors: figure out who your helpers are
- Mindset: you're in it for the long haul, so get your ship in shape

Nail down your process

- Your docs should be complete and clear to a newcomer
  - Get someone 'new' to try and get quickstart your project based on them
- Your CI should be comprehensive & rock solid
  - Regressions are expensive, contributing should be easy
- Clear PR strategy & convention

Figure out how to sustain yourself

- Be an authentic version of yourself in public. It's far less tiring than a facade
- Be honest with yourself about where your emotional gaps are & don't try to be someone you're not
- This is the oldest advice in the book but also 100% true
- Don't do anything you don't want to. There's a lot of room out there

fin