THE JOY OF OPEN SOURCE SOFTWARE

An introduction to the world of shared workforce

OUTLINE

This presentation is meant to provide key insight on how open source software development works, and how to (humbly) give back.

- The contract of OSS
- What we need by “it works” ?
- So you found this problem…

PREREQUISITES

1. git (command line)
2. Basic notions of testing

OUT OF SCOPE

Proposing and Contributing features (new code)

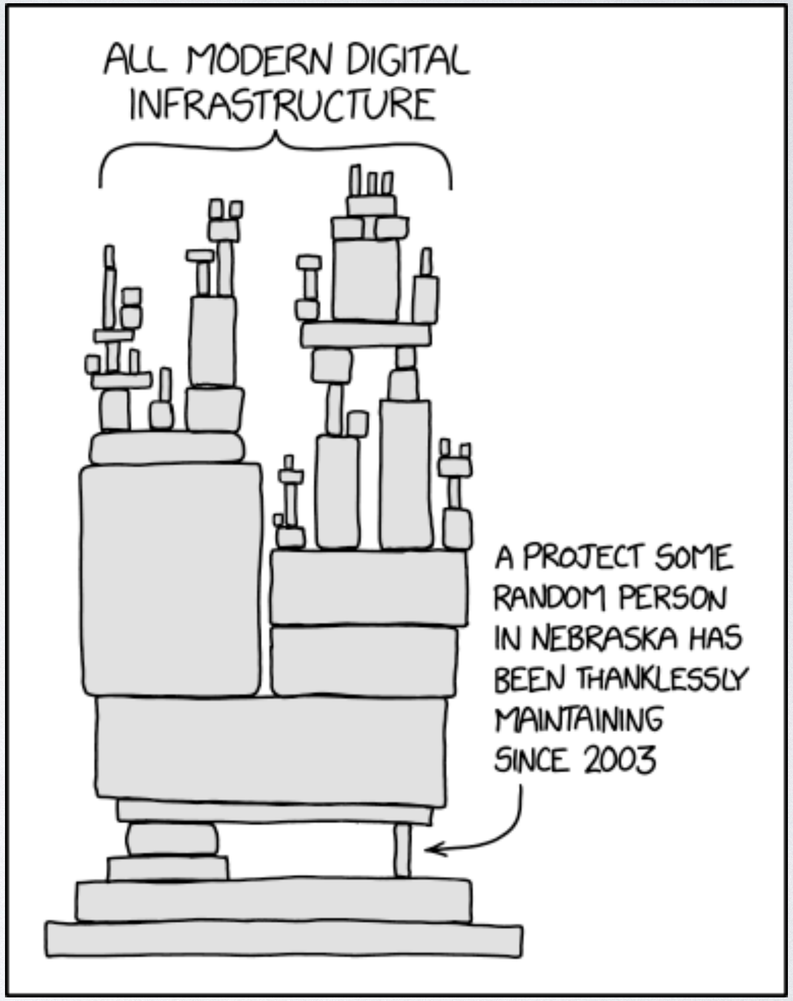
THE CONTRACT OF OSS

What Open Source Software means

1. Openly accessible
2. Free for use, depending on LICENCE
3. May rely on (or accept) volunteer contribution

What it's often interpreted as

1. Free resources
2. Free labour (maintenance)



https://xkcd.com/2347/

THE CONTRACT OF OSS

Contributing is not just writing code !

You can help by providing feedback, reporting issues, proof reading documentation, writing tests …

Maybe you spent 1''30' writing this script and realised that it would have been trivial if you knew how to use this function ?

Help others by making their experience less miserable than your own, improve documentation !

Maybe you just updated this library and it broke your application code ?

Tell maintainers so they can fix it !

SIMPLE CONTRIBUTION EXAMPLES

Good:

- adding a missing docstring: https://github.com/yt-project/yt/pull/2056
- fixing a simple bug: https://github.com/yt-project/yt/pull/2366

Bad:

- refactoring with no clear motivation: https://github.com/tiangolo/fastapi/pull/5299
- formatting someone else's code: https://github.com/yt-project/cmyt/pull/85

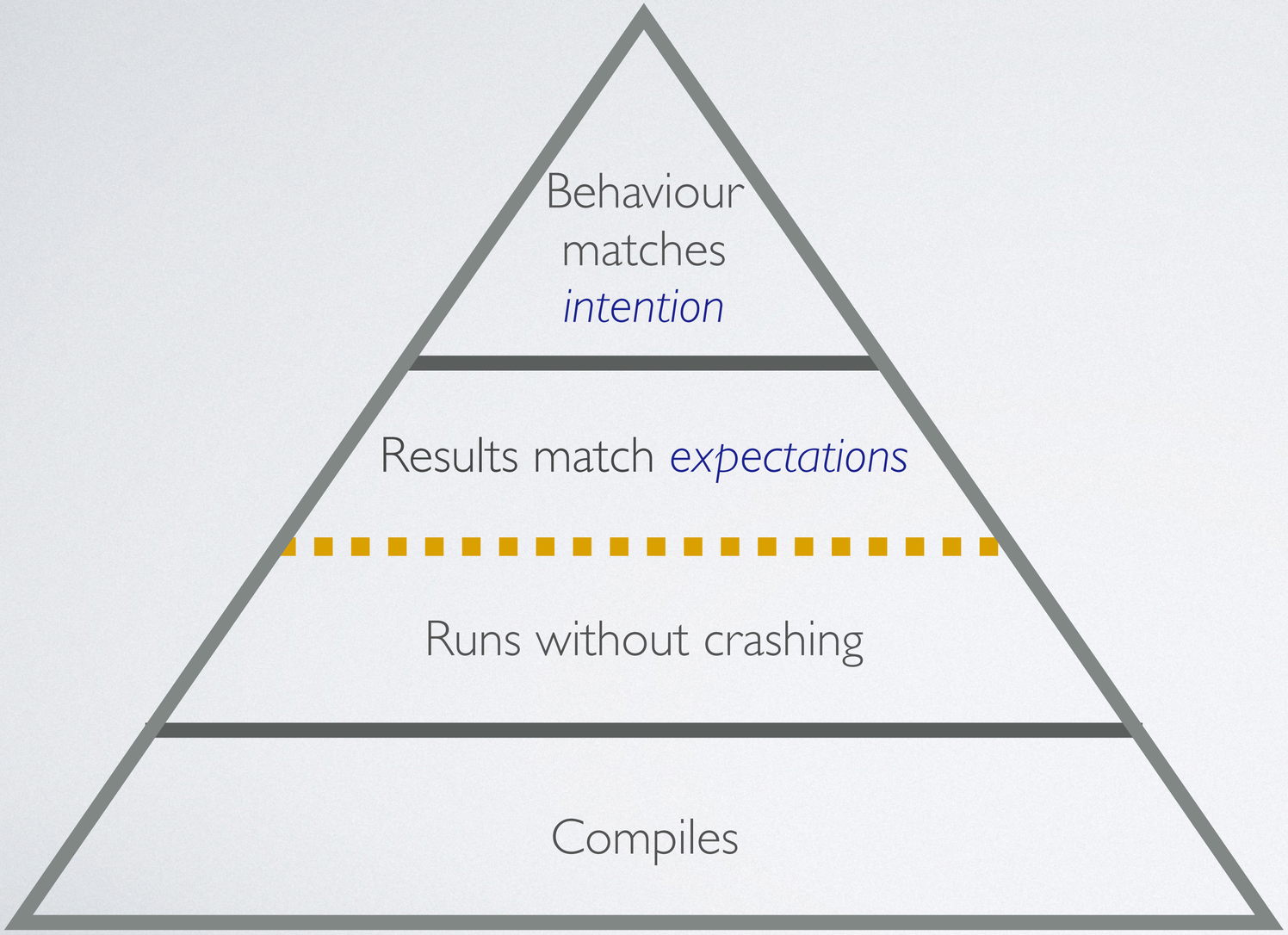
WHAT WE MEAN BY "IT WORKS"

In many cases, code (or documentation) is obviously™ broken

1. Compilation error
2. Runtime error (crash)
3. Unreasonable results
4. Typos in docs

Q: These are all clear opportunities to contribute, but how else can you tell something is buggy or needs improvements ?

WHAT WE MEAN BY "IT WORKS"



Abstract* (subjective)

Concrete (objective)

WHAT WE MEAN BY "IT WORKS"

Abstract requirements can be made concrete with explicit declaration of intention

1. Docs
2. Tests
3. Possibly a discussion thread

A bug can be any mismatch between intended and observed behaviour

WHAT WE MEAN BY "IT WORKS"

Q: So … how can you tell what the intention is ?

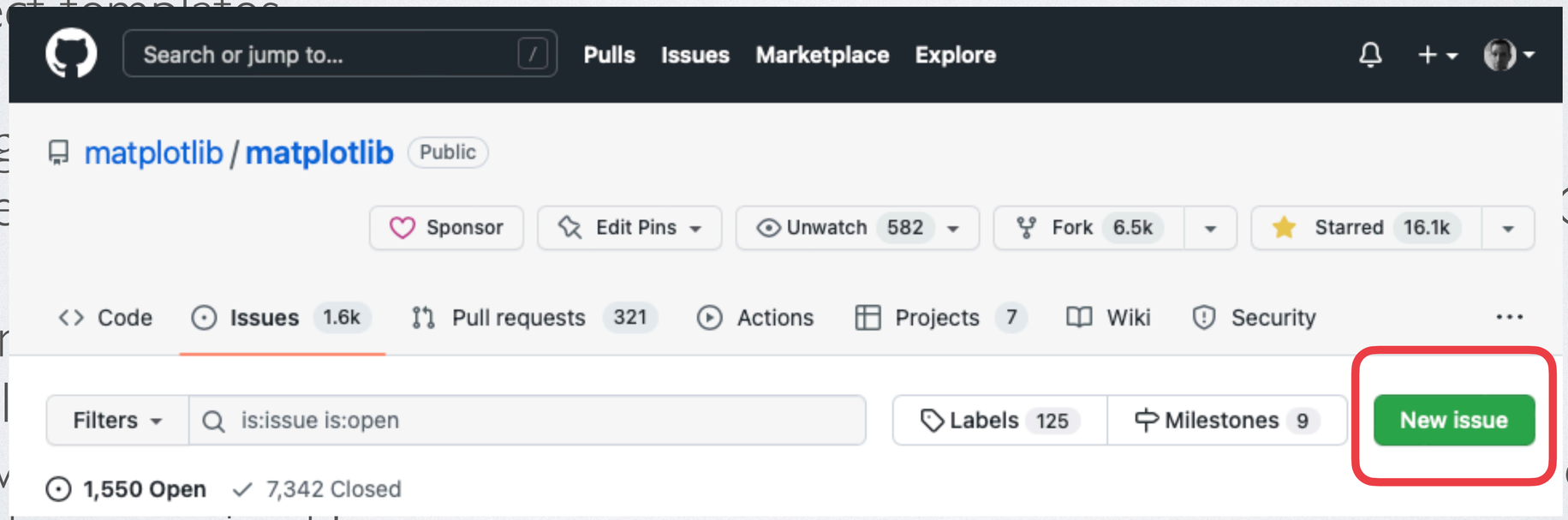
1. Search the docs
2. Did behaviour change ? Search release notes (aka changelog)
3. Search for an existing question or ticket
   (Github ? SlackSpace ? Mailing list archives ?)
4. If all fails, ask in any relevant channels (search for guidelines)

SO YOU FOUND THIS PROBLEM…

I - Say it loud !

Problems don't solve themselves, someone in charge needs to know about it.
Often, this means you need to file a report on the issue tracker (most likely Github).
Here are some "universal" guidelines to do it in a useful and efficient fashion

1. Respect templates
2. Write for solving
3. Be spe On which
4. Be hur
5. Be pol
6. Follow cket might have received !



SO YOU FOUND THIS PROBLEM…

2 - Provide a solution !

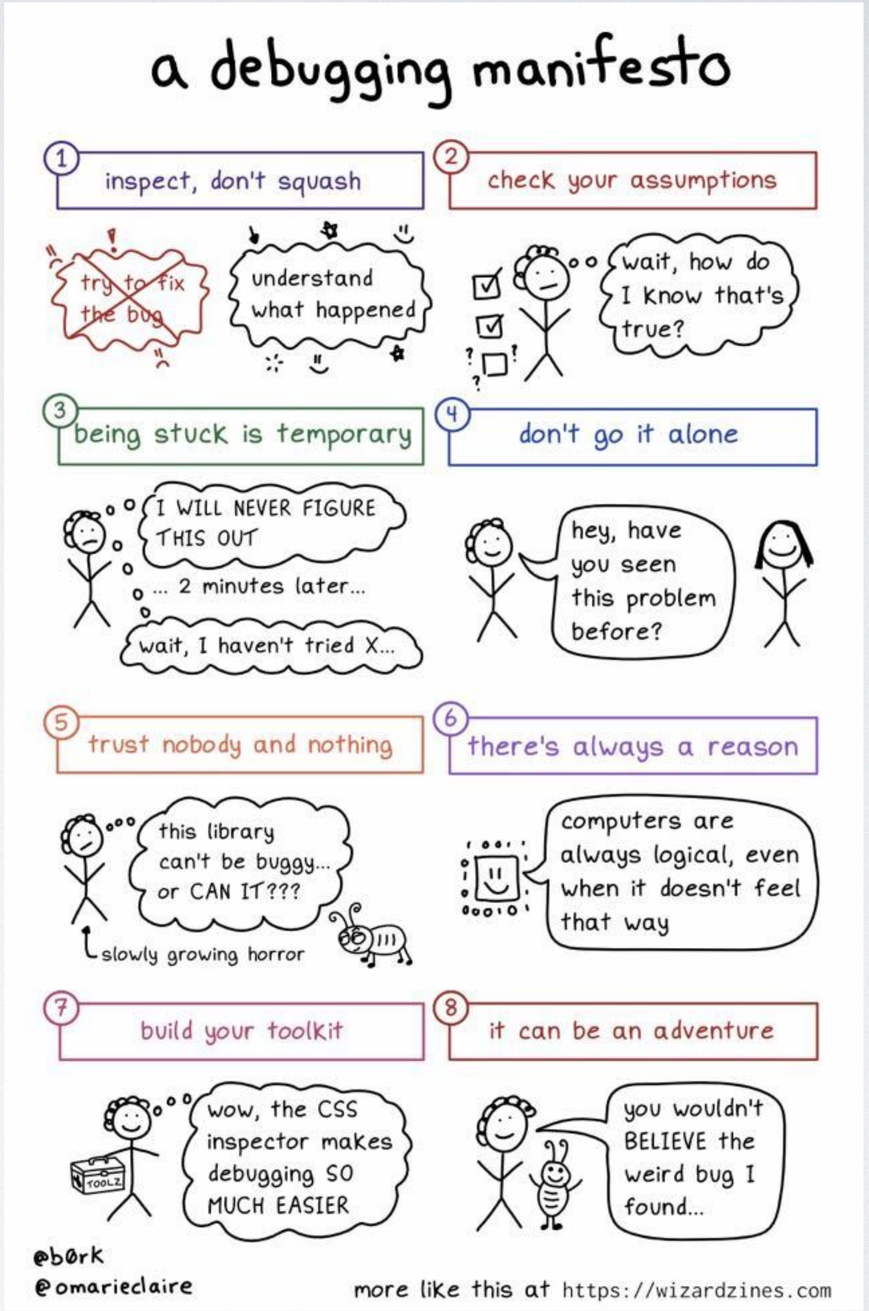
Some problems don't require a lot of expertise to resolve. In general, big projects have contributor guides, but here are some "universal" guidelines on how to proceed.

1. Fork the project
2. Clone your fork
3. (Create an isolated environment dedicated to this project)
4. Install your local copy so that you can reproduce the issue in isolation
5. Write a test that shows the problem ?
6. Understand what's happening (navigate and question the code you're using)
7. Change the code until your test pass
8. Make any required changes so that other existing tests pass
9. Add your changes and your test(s) to a branch
10. Push that branch to your fork
11. Open a Pull Request to the main repo

TAKE HOME

Don’t stay silent: problems don’t solve themselves
Don’t deify code you are using: everyone makes mistakes
Be humble: assume you may have made a mistake

BONUS SLIDE


a debugging manifesto
1 inspect, don't squash
try to fix the bug
understand what happened
2 check your assumptions
wait, how do I know that's true?
3 being stuck is temporary
I WILL NEVER FIGURE THIS OUT
... 2 minutes later...
wait, I haven't tried X...
4 don't go it alone
hey, have you seen this problem before?
5 trust nobody and nothing
this library can't be buggy... or CAN IT???
slowly growing horror
6 there's always a reason
computers are always logical, even when it doesn't feel that way
7 build your toolkit
TOOLZ
wow, the CSS inspector makes debugging SO MUCH EASIER
8 it can be an adventure
you wouldn't BELIEVE the weird bug I found...
@b0rk
@omarieclaire
more like this at https://wizardzines.com