How to contribute to (Scala) open source & my experience

Jimin Hsieh

Agenda

- Me
- Definition
- How to contribute to open source?
- Why?
- Step by step
- Type of contributions
- How to find the projects or issue?
- GitHub Search
- How to send the PR
- Maintainers
- Mentality

Me

- Jimin Hsieh
  - Close to 150 PR at GitHub from 2017/01
    - Scala open sources is the most I contributed to.
    - After removing the owners of repositories I know, it will be 14x PR at GithHub.
  - Member of Scala GitHub organization
  - English is not my mother tongue and I am not good at it too.
  - I don’t get paid for contributing to open source.

Definition

- Open source
  - Open source is a licensing and delivery mechanism, period. It means you get the source for software and the right to use and modify it. From Rick Hickey
  - It could be owned by a commercial company.

How to contribute?

How to contribute?

I mean it. You just have to have your first move.

Why

- Learning new things
- Implement what you learn
- **You got the code review by**
    - **Experienced developers**
    - **Maintainers or committers**
    - **Java Champion**
- List your contributions at your resume
    - Most employers won't take a look in Taiwan.
- Fun
- Make the world a little better

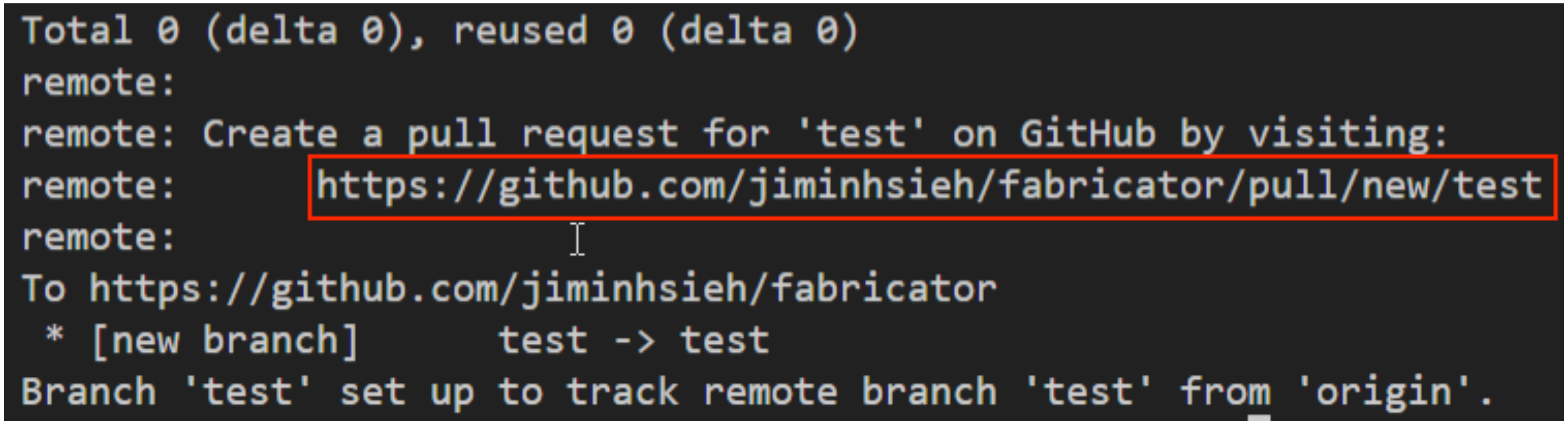
How to do it step by step?

- Fork project to your side
- Clone the repository from your side to local
- Create your working branch
- Add your patch
- Push your working branch from local to your remote

Add your patch

- Git commit messages from OpenStack
- 7 rules of a great Git commit message from Chris Beams
- Each commit should just **do just one logical change**.

Push to your remote

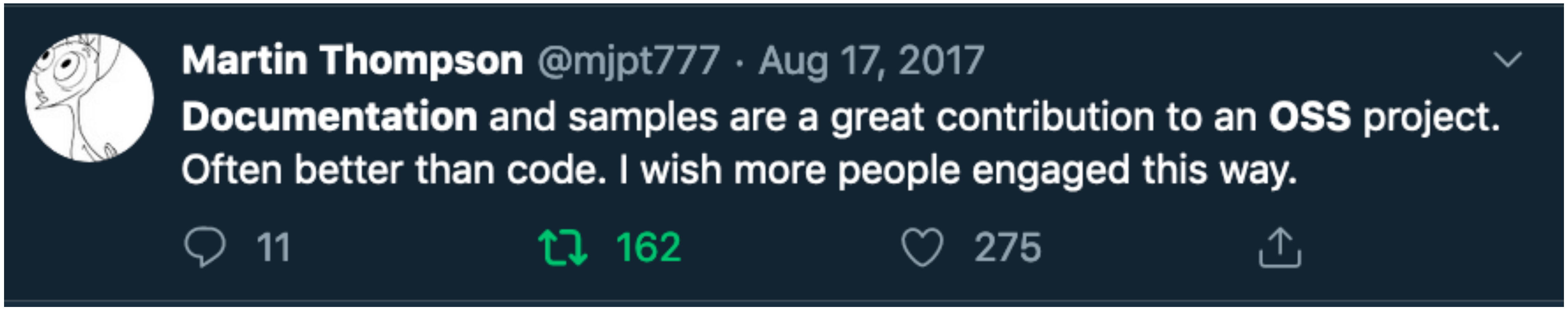


```
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'test' on GitHub by visiting:
remote:      https://github.com/jiminhsieh/fabricator/pull/new/test
remote:
To https://github.com/jiminhsieh/fabricator
 * [new branch]      test -> test
Branch 'test' set up to track remote branch 'test' from 'origin'.
```

Type of Contributions

- Documentation
- CI/CD
- Deprecate
- Upgrade
- Add unit tests
- Fix Bugs
- Features
- Raise issues
- Help code review
- Join the discussion

Documentation


Martin Thompson @mjpt777 · Aug 17, 2017

Documentation and samples are a great contribution to an **OSS** project. Often better than code. I wish more people engaged this way.

11 162 275

What you can do for documentation?

- Typo
- The explanations for new features
- The examples implementation for new features

CI/CD

- Bad CD is bad. The only bad can happen is you can not deliver your shitty software.
- Bad CI is really bad for the software and maintainers.
  - False positive.
- You will need to learn your build tool, Travis CI, and CirecleCI.
  - You probably need GitHub Actions later

What you can do for CI/CD?

- Cache libraries
- Source code format check
- Unit Testing & Test Coverage
- Static code analyzer
- Delivery to library repositories
    - It will a little harder. You need to mimic how the maintainers publish their libraries.

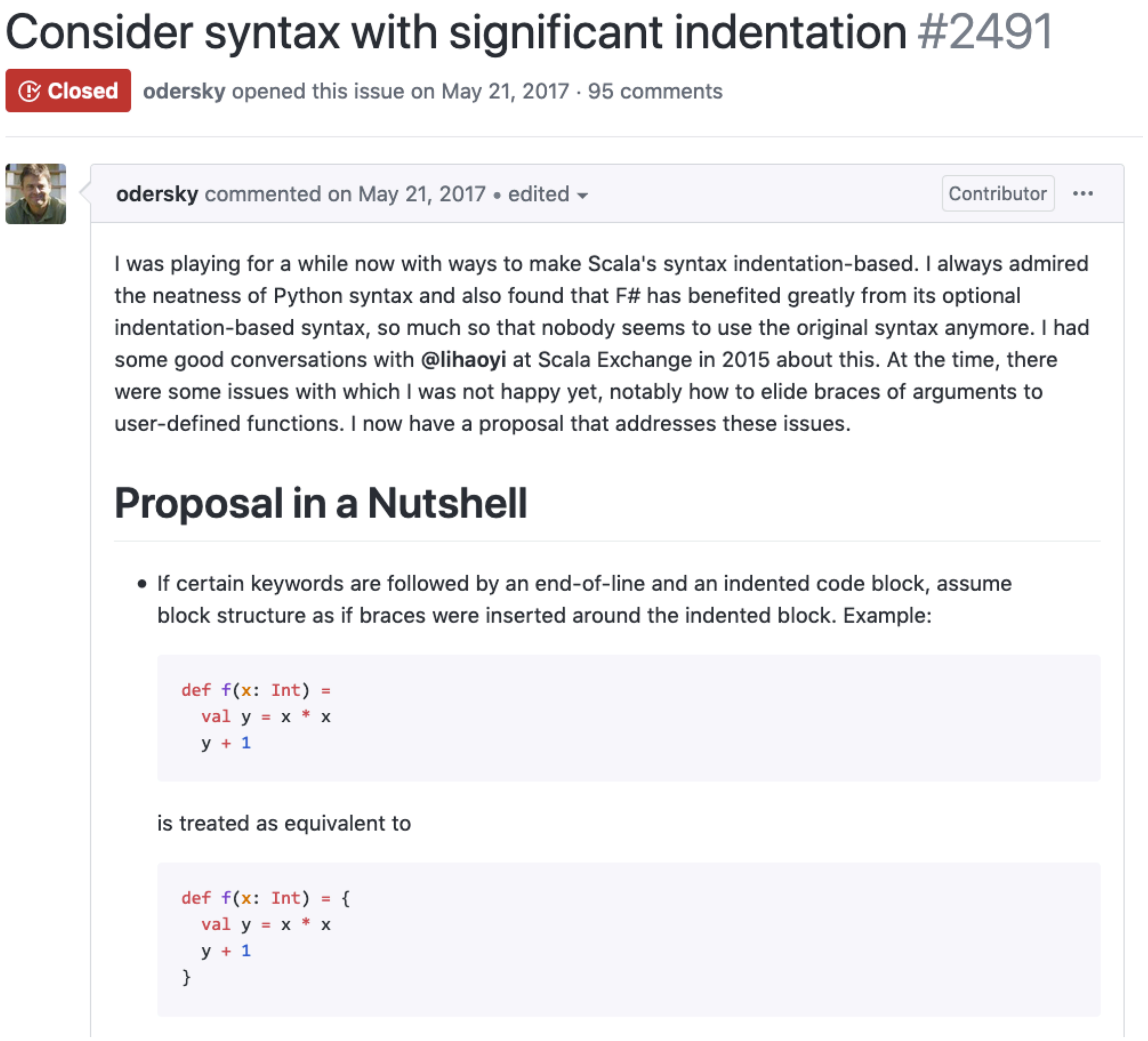
Deprecate, Upgrade, Unit Tests, Bugs

- You can contribute the code with a clear goal.

Features

- It will be the hardest part of contributions since the expectation is not that clear.
  - You will need a lot of discussions.
- 2 types of features
  - From the maintainers
  - From your own

Join the discussion



Consider syntax with significant indentation #2491

Closed **odersky** opened this issue on May 21, 2017 · 95 comments

odersky commented on May 21, 2017 • edited

I was playing for a while now with ways to make Scala's syntax indentation-based. I always admired the neatness of Python syntax and also found that F# has benefited greatly from its optional indentation-based syntax, so much so that nobody seems to use the original syntax anymore. I had some good conversations with **@lihaoyi** at Scala Exchange in 2015 about this. At the time, there were some issues with which I was not happy yet, notably how to elide braces of arguments to user-defined functions. I now have a proposal that addresses these issues.

Proposal in a Nutshell

- If certain keywords are followed by an end-of-line and an indented code block, assume block structure as if braces were inserted around the indented block. Example:

```
def f(x: Int) =
  val y = x * x
  y + 1
```

is treated as equivalent to

```
def f(x: Int) = {
  val y = x * x
  y + 1
}
```

Raise issues

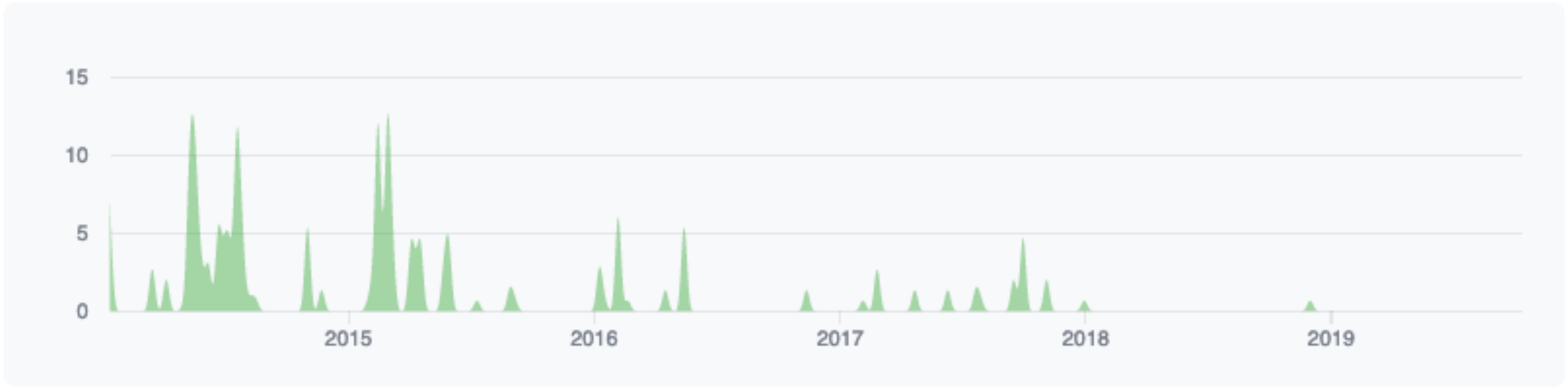
- Propose new ideas
- Find any bugs
    - The maintainers should be able to **reproduce the issues** from your comments.

How to find the projects or issues?

- Find the projects you’re using at current job.
- Find the projects you’re interested.
- Don’t pick up the projects were archived.
- Check out GitHub repository insights.

Check out repository insights

Contributions to master, excluding merge commits



Still no ideas?

- good first issue
  - Scala & Java
- help wanted
  - It contains different levels of issues.
  - Scala & Java

- Scala Spree
- Scala Main Repository
- Scala Center
- GitHub Trending

How to use GitHub search?

- The magic key word for GithHub search
  - is:open is:issue label:"good first issue”
  - is:open is:issue label:"help wanted”
  - is:open is:issue label:"low hanging fruit"

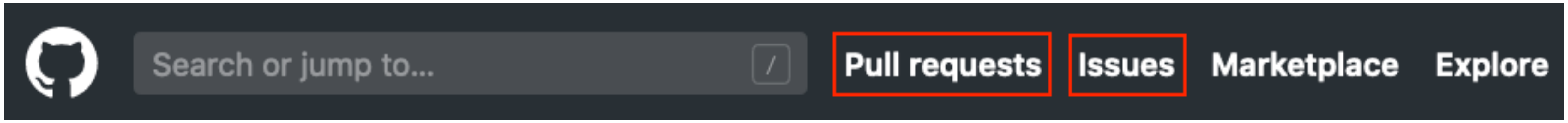
How to send the PR?

- Check `CONTRIBUTING.md`
- Try to understand the issue.
- If you have any questions, leave the question at the comment.
  - Most of the maintainers are good.
- Send a small/obvious PR to the project to test how’s it going.
  - **Some of projects almost die. Maintainers don’t archive it nor respond to any issues or PRs.**

After the PRs

- Some of people prefer the original commits. Some of people prefer you squash those commits per features or one.
    - Discuss with your maintainers. Ask what’s their preference.

Manage the PR or Issues



Maintainers

- **Be patient.**
  - Don’t expect they will reply you soon.
    - They have lots of mails. They probably are busy on other things.
- **Be nice and polite.**
  - They don’t own you anything. Most open source maintainers don’t get paid for open source.
  - Meanwhile, you don’t own them anything as a contributor.

Mentality

- You are not your code.
- Expect you will make a mistake.
- Read, read, and read. Then ask or ask.
    - Read the books, read the source code, and read the documentation. Then ask at discussion channel (IRC, Slack, Gitter…etc) or ask at GitHub issues.

Reference

- Contributing to Open Source: A Guide - Paul Ganssle

Thanks for your coming.

Any question?