Developing and Maintaining Open Source Software

...

Speakers:

- Ioannis Christodoulou
- Ioannis Kapolos
- George Kiagiadakis
- Foivos Zakkak

TODO

Daknob promised

“In this topic, we will see how we can start a new project and Open Source it on GitHub, Open Source an existing project we may already have, contribute to an already existing project, as well as maintain one once we get started having Pull Requests.”

-- https://github.com/devstaff-crete/DevStaff-Heraklion/issues/144

Contributing to Open Source Projects

Ioannis Christodoulou

Why?

Where do I start from?

Finding a *suitable* project

Identifying how to contribute

Contributing

What's next?

Why should I even bother?

Improve your skills

- coding
- UI
- Documentation
- social

Meet people with similar interests

Grow a reputation, demonstrate your skills

Where do I start?

Find a project you want to work on

1. Projects you currently work with
2. Projects you would be *excited* to work with
3. Projects welcoming contributions:
    a. opensourcefriday.com
    b. firsttimersonly.com

Project suitability

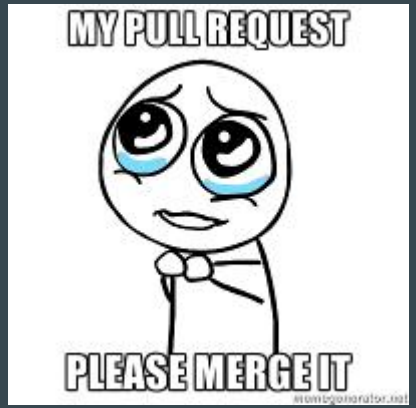
1. Open Source License
2. Active
3. Issues
4. Pull requests
5. Welcoming maintainers

Identifying how to contribute

1. README / CONTRIBUTING
2. Communication
    - *Watch* the project (issues / PRs)
    - Check project's chat (Gitter / Slack / mailing list / Google group / etc)
3. Tips for effective communication
    - Do your homework
    - Short and clear context
    - Keeping it public and respectful
    - Issues for errors, features, ideas
    - Directly PRs for typos, obvious errors

Contribute

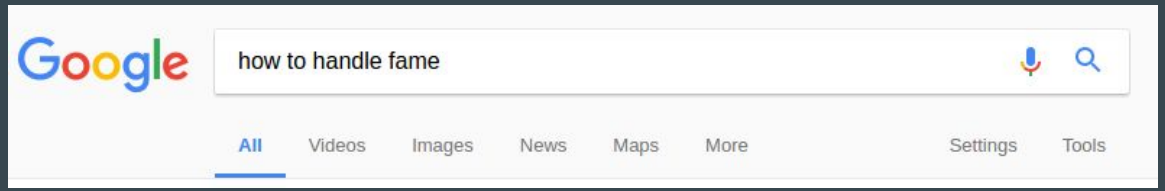
1. Communicate what you are working on
2. Get the project running locally
3. Write your code
4. Follow code style
5. Test your changes!
6. Pull request



What's next?

Possible outcomes

- Nothing
- Changes requested
- Accepted!



Maintaining Open Source Software

Foivos Zakkak

Why bother?

Interacting with people

Accepting contributions

Code review

Maintaining the quality standards

Keep it alive

Why bother?

It's fun!	Not that much
Someone has to do it...	True
Feeds your ego	Yes, but don't
Give back to the community	Good samaritan
Get paid for it!	Definitely
Cause reasons...	Always...

Interacting with people

Who?

- Users
- Bug reporters
- Developers / Contributors

How?

- Documentation
- Issues / Bug reports
- Mailing list
- Chat (slack, IRC, etc.)
- Personal e-mail

Rules:

1. Be polite (no matter what)
2. Be patient
3. Be supportive
4. Be encouraging
5. Acknowledge
6. Don't rush
7. Don't forget that everything is often public
   (shouldn't matter anyway)

Accepting contributions

Channels:

- **Pull / Merge requests**
  Provided by online tools like github, bitbucket, gitlab etc.
- **Patches / Diffs**
  Usually sent through emails, not that common nowadays
- **Comments / Ideas**
  Through discussions on chat rooms, mailing lists, issues, and bug reports

Filtering:

1. Does it introduce something useful to the many or just the contributor?
2. Does it make sense?
3. Is it generic?
4. Does it break someone's workflow?
5. Does it agree with the concept and principles of the project?

Code review

What?

- Read the diffs / patches
- Look for errors
- Look for deficiencies
- Make sure documentation is also updated
- Propose / Ask for improvements where needed

Don't:

- feel bored, it's your duty
- skip large chunks of code
- be afraid you will offend the contributor (just be polite)

DO:

- Hack it yourself if needed (some contributors might not be as experienced as you)

Maintaining the quality standards

Automate things:

- Provide tests (e.g. `make test`)
  - Unit Tests
  - Integration Tests
- Use tools for style-checking (e.g. `checkstyle`)
- Use Continuous Integration (CI) (e.g. travis, jenkins)
- Use git hooks?

Avoid breaking backwards compatibility

- Consider using semantic versioning (http://semver.org/),
  i.e. `MAJOR.MINOR.PATCH`

Keep it alive

- Promote it
  - Have a nice clean front page for the project
  - Add a demo of what it does on youtube
  - Include screenshots in the front page if applicable
  - Social Media
  - Related mailing lists, Websites
  - MeetUps
  - ...
- Be active and responsive
- Find people to continue your work
  - You can't do it forever

OSS for fun and $profit

Tales from the dark side

The big picture

OSS is a fun thing to do

OSS is a benefit for the consumers (that's you)

OSS is a benefit for the producers (that's you again)

OSS is not just about coding

You don't need to be a ninja rockstar dev to produce value

The Real FUN of OSS

Yes, when you build something you like, the process itself is fun!

Yes, when you work in a community setting, it is fun!

But the real fun is...

When people you don’t know heartily suggest your product to others.

The 2 currencies of OSS

Code

Value you add

Community

Value you bring

How do you leverage them?

Leverage

6 Strategies for the Individual

Things that I've done and/or seen other people doing

Leverage Strategy I

Difficulty Level: Medium

The Starry Sky

The more you have
the more it shines

Stars - what are they?

How many do I need?

What project should I choose?

How do I acquire them?

Leverage Strategy II

Difficulty Level: Easy

The Fishnet

Throw a wide net,
Catch a Whale

Repo all the things!

No need for perfection

The more diverse, the better

Examples

Leverage Strategy III

Difficulty Level: Easy

The Power User

I use therefore I am

Report bugs

Make feature suggestions

Triage issues

Examples

Leverage Strategy IV

Difficulty Level: Easy

The IN Crowd

I'm soooo like you,
Let's work together

Tons of jobs get filled BEFORE they reach the usual "looking for ninja rockstar" places

Hang out with the community

Slack & IRC work best

Speak when you know what you're talking about, otherwise let people assume you know

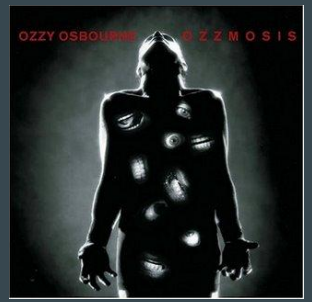
Examples

Leverage Strategy V

Difficulty Level: Medium

I’m with Her

Social Proof by Osmosis



Find the movers in your target community

Meaningfully contribute to their pet project(s)

Now you have neat references (with their permission, don’t be an ***)

Examples

Leverage Strategy VI

Difficulty Level: Easy

The Scholar

Behind every great project,
Lies the absence of meaningful documentation

Big projects have documentation teams and few people in them

Exchange manual, boring labor for becoming the newest member of OMGLib/Doc team

Most people simply assume you are an expert - they don't let just anybody become part of the OMGLib team, right?

Unadvertised Bonus!

Difficulty Level: Medium

Deus Ex Machina

You Batman, you

Project authors lose interest or life gets in the way

Many times, they want someone to take over

Take over, what are you waiting for?

Examples

2 Strategies for Small Companies

Leverage Strategy VII

Difficulty Level: Really Hard

Dual Licensing

Take this for free!
Oh, you want a cluster?
$40.000/year/server

Create something of big value

Give it away for free for any kind of use (i.e. let people make money)

Get raving Evangelists

Charge a bazillion dollars for Enterprise features (and support)

Examples

Leverage Strategy VIII

Difficulty Level: Hard

Hipster Power

Here's the coolest new microframework!
I've re-invented the wheel,
But it is PURPLE!!!!!!!!!!

The rate of change for many parts of the industry is absurd

Create a good framework around a bleeding-edge hipster niche buzzword

Create a hip site about it (you want to pay a hip designer for this)

They flock, you get bragging rights

Works with non-hipster stuff too ;)

Examples

Outro

The industry in FOSS

Giorgos Kiagiadakis

$ whoami

FOSS contributor since 2008

Google Summer of Code, 2009

Senior Consultant Software Engineer, Collabora Ltd., 2010 - present

Consultancy?

Enabling industry leaders to develop product solutions using FOSS

Development guidance

Bug fixing

Hardware integration

Packaging / Distribution maintenance

Training

Open sourcing proprietary code

Why FOSS?

Reduce development, testing & maintenance costs

Reduce development time

High quality code

Developer friendly

Is it profitable?

Sure!

Hardware sales

Service sales

Product support / warranty

Consulting

Licensing

Welcome to the dark side!

FOSS is copyrighted!

Licenses are applied to grant freedoms

```
/*
 * Copyright (C) 2017 George Kiagiadakis <gkiagia@tolabaki.gr>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */
```

Exception: software in the public domain

Which freedoms?

Free redistribution (source & binary)

Source code access

Derived works / modification

...

The license minefield

GPL, LGPL, AGPL

BSD 3-clause, BSD 2-clause

MIT, Apache

MPL, CDDL, Eclipse PL

...

Copyleft

Restricting freedom

Give modifications to the users

No license change or change to compatible license

No linking to more “closed” software (workaround with a plugin)

Not linked to by more “closed” software (GPL)

Grant a patent license (GPLv3)

Challenges

If you are unsure, talk to your lawyer

Combining software with copyleft

Patent licenses (not always granted, or forced to be granted)

Copyright assignments

License violations