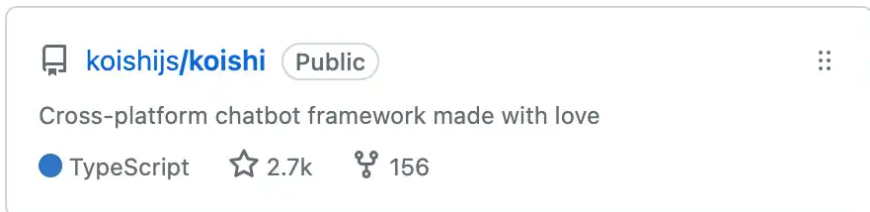
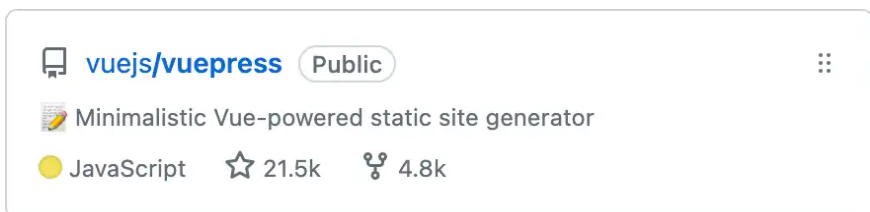
开源软件

自我介绍

https://github.com/shigma





小调查

- 有多少人使用过 GitHub?
- 你们认为开源软件是什么?
- 你知道哪些开源软件?

开源是什么？

Open Source Software (OSS / FOSS)

- 源代码可以任意获取
- 以分散、协作的方式开发
- 并不一定指软件

有哪些开源软件？

- Linux (Debian, Ubuntu, …)
- Git
- MySQL
- VSCode
- Firefox
- WordPress
- Kubernetes
- ……

巨头也在开源

- Google
  - TensorFlow
  - Android
  - Chromium
- Facebook
  - React
  - PyTorch
- Microsoft
  - .NET
  - TypeScript
  - Visual Studio Code

从自由软件到开源软件

~~我免费了！~~

- 自由软件是最先提出的概念 (1985)
  - 反义词：专有软件
- 自由软件 (free) 一定是免费 (free) 的吗?
- 自由软件一定不能商业化吗?
- 后来出现了开源软件的概念 (1998)
- 现在也有 FOSS 的说法 (Free and Open Source Software)

参考：

- 自由软件基金会 (FSF): https://fsf.org/
- 开放源代码促进会 (OSI): https://opensource.org/

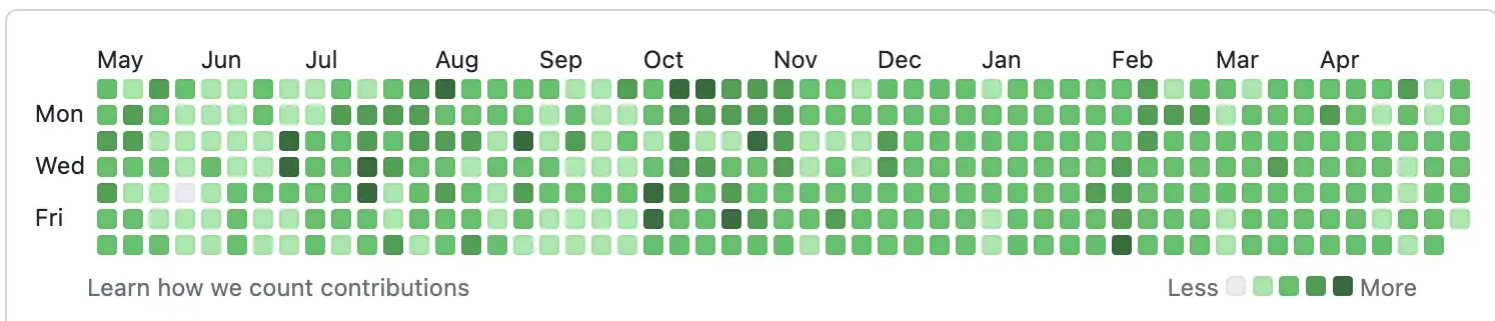
开源软件的价值

为什么人们要放弃专有而选择开源?

- 同行评审
- 透明性
- 可靠性
- 灵活性
- 更低成本
- 无供应商锁定
- 开放式协作

参与开源，我们能获得什么？

- 参与自己感兴趣的项目，在充实中度过大学生活
- 提高知识水平，深入了解软件的运行原理
- 提高工程能力，问题分析和解决能力
- 积累项目管理经验，人际交往能力
- 开源经历本身也是大企业的敲门砖



参与开源，我们能做什么？

- 探索别人写好的项目
  - 在使用中遇到了问题
    - 提 issue 本身也是一种贡献！
  - 如果有能力，可以尝试修复问题
    - 向项目提交 Pull Request
  - 增加新功能，让项目变得更好
- 自己开发项目，甚至可以是
  - 课程作业
  - 个人博客

贡献文档

这并不丢脸 (笑)

- Typo Fix
- 国际化 / 翻译

我自己的例子:

- https://github.com/nodejs/node/pull/34980
- https://github.com/nodejs/node/blob/main/doc/changelogs/CHANGELOG_V14.md

client	feat(client) : add `draw.io` (#438)	3 weeks ago
dist	chore(release): 0.8.19	2 weeks ago
docs	Update extensions.CN.md	8 months ago
examples	feat: #424 增加对齐方式语法和对应按钮	3 weeks ago
logo	chore(*): normalize eol	10 months ago
src	chore(release): 0.8.19	2 weeks ago
test	feat: update unit test (#188)	last year
types	feat: #412 增加textarea的name属性配置能力	3 weeks ago
vscodePlugin	feat: vscode 插件支持预览工作区图片	last month
.code.yml	✨ init repo	2 years ago
.cz-config.js	✨ init repo	2 years ago
.editorconfig	✨ init repo	2 years ago
.eslintignore	feat: 增加vscode plugin，可以在vscode扩展中输入cherry-markdown...	3 months ago
.eslintrc.js	feat: add eslint jest plugin	2 years ago
.gitattributes	feat(eslint): setup husky and lint-staged	10 months ago
.gitignore	refactor : refactor client (#428)	3 weeks ago
.gitpod.yml	docs(examples): fetch markdown files directly instead of load with o...	2 years ago
.prettierignore	docs(*): add examples	2 years ago
.prettierrc.js	chore(*): lint fix	last year
.versionrc.json	✨ init repo	2 years ago
.yarnrc	feat: add yarnrc & update dependencies	2 years ago
CHANGELOG.md	chore(release): 0.8.19	2 weeks ago
LICENSE	✨ init repo	2 years ago
README.CN.md	feat: (hooks) 自动超链接-支持展示固定长度字符 (#391)	3 months ago
README.md	feat: (hooks) 自动超链接-支持展示固定长度字符 (#391)	3 months ago
babel.config.js	feat: support more code highlight (#347)	6 months ago
gulpfile.js	✨ init repo	2 years ago

开源项目里有什么

源码仓库中有许多目录，它们分别是做什么的呢？

- art/logo：项目的 logo
- build/script：存放构建脚本
- client：客户端
- dist：生成的文件 (distribution)
- docs：项目文档
- examples：示例
- lib：库 (library)
- packages：包
- src：源代码 (source)
- test：测试用例
- types：类型定义

client	feat(client) : add `draw.io` (#438)	3 weeks ago
dist	chore(release): 0.8.19	2 weeks ago
docs	Update extensions.CN.md	8 months ago
examples	feat: #424 增加对齐方式语法和对应按钮	3 weeks ago
logo	chore(*): normalize eol	10 months ago
src	chore(release): 0.8.19	2 weeks ago
test	feat: update unit test (#188)	last year
types	feat: #412 增加textarea的name属性配置能力	3 weeks ago
vscodePlugin	feat: vscode 插件支持预览工作区图片	last month
.code.yml	✨ init repo	2 years ago
.cz-config.js	✨ init repo	2 years ago
.editorconfig	✨ init repo	2 years ago
.eslintignore	feat: 增加vscode plugin，可以在vscode扩展中输入cherry-markdown...	3 months ago
.eslintrc.js	feat: add eslint jest plugin	2 years ago
.gitattributes	feat(eslint): setup husky and lint-staged	10 months ago
.gitignore	refactor : refactor client (#428)	3 weeks ago
.gitpod.yml	docs(examples): fetch markdown files directly instead of load with o...	2 years ago
.prettierignore	docs(*): add examples	2 years ago
.prettierrc.js	chore(*): lint fix	last year
.versionrc.json	✨ init repo	2 years ago
.yarnrc	feat: add yarnrc & update dependencies	2 years ago
CHANGELOG.md	chore(release): 0.8.19	2 weeks ago
LICENSE	✨ init repo	2 years ago
README.CN.md	feat: (hooks) 自动超链接-支持展示固定长度字符 (#391)	3 months ago
README.md	feat: (hooks) 自动超链接-支持展示固定长度字符 (#391)	3 months ago
babel.config.js	feat: support more code highlight (#347)	6 months ago
gulpfile.js	✨ init repo	2 years ago

根目录下的文件

配置文件通常以点 (.) 开头，我们称为 dotfile。

> 这样的文件在 Linux 系统中是隐藏的。
> 我们可以通过 `ls -a` 命令查看。
> 此外，形如 *.config.* 的文件通常也是配置文件。

以大写字母为名称的文件：

- README.md：项目介绍
  - README.CN.md：中文介绍
- CHANGELOG.md：更新日志
- LICENSE：许可证
- CONTRIBUTING.md：贡献指南
- CODE_OF_CONDUCT.md：行为准则
- SECURITY.md：安全策略

许可证

开源许可证是开源软件生态系统的基础，可以促进软件的协同开发。

- MIT
- Apache
- BSD
- GPL / AGPL / LGPL
- CC BY-SA
- ……

MIT

Permissive 与 Copyleft

- Permissive
  - 对软件的使用、修改、传播等方式采用最低限制
  - 为原作品的自由使用、修改和传播等提供更大的空间
  - 典型代表：MIT，Apache，BSD
- Copyleft / “著佐权”
  - 自由软件必须将自由无限传承下去
  - “传染性”——衍生作品需要遵循相同条款
  - 典型代表：GPL，AGPL，LGPL

Licenses

Open source licenses grant permission for anybody to use, modify, and share licensed software for any purpose, subject to conditions preserving the provenance and openness of the software. The following licenses are sorted by the number of conditions, from most (GNU AGPLv3) to none (Unlicense). Notice that the popular licenses featured on the **home page** (GNU GPLv3 and MIT) fall within this spectrum.

If you're looking for a reference table of every license on choosealicense.com, see the **appendix**.

GNU AGPLv3

Permissions of this strongest copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights. When a modified version is used to provide a service over a network, the complete source code of the modified version must be made available.

Permissions

- Commercial use
- Distribution
- Modification
- Patent use
- Private use

Conditions

- Disclose source
- License and copyright notice
- Network use is distribution
- Same license
- State changes

Limitations

- Liability
- Warranty

View full GNU Affero General Public License v3.0 »

知识共享许可协议

Creative Commons (CC) License

- BY：署名
- SA：相同方式共享
- NC：非商业性使用
- ND：禁止演绎

一共有多少种组合方式?

- CC BY
- CC BY-SA
- CC BY-NC
- CC BY-NC-SA
- CC BY-ND
- CC BY-NC-ND

参与开源，要学习哪些基础知识?

- Git
- Markdown

Markdown

Markdown

Markdown 是一种轻量级标记语言，它可以让我们更加方便地书写文档。

- 项目文档和 README
- 项目的 issue 和 pull request
- 各种聊天软件 (Discord, Slack, ...)
- 各种效率软件 (Notion, Obsidian, 语雀, 飞书, 钉钉, ...)
- 各种博客平台 (WordPress, Hexo, VitePress)
- 你现在看到的幻灯片 (slidev)

标题

```
# 一级标题
## 二级标题
### 三级标题
#### 四级标题
##### 五级标题
###### 六级标题
```

一级标题

二级标题

三级标题

四级标题

五级标题

六级标题

字体

```
**这是加粗的文字**
*这是倾斜的文字*
***这是斜体加粗的文字***
~~这是加删除线的文字~~
```

这是加粗的文字

这是倾斜的文字

这是斜体加粗的文字

~~这是加删除线的文字~~

引用

```
> 这是引用的内容
```

> 这是引用的内容

图片与链接

```
![图片](https://koishi.chat/logo.png)

[链接](https://koishi.chat)
```

链接

列表

```
- 无序列表
  - 二级列表
    - 三级列表

1. 有序列表
2. 有序列表
3. 有序列表
```

- 无序列表
  - 二级列表
    - 三级列表

1. 有序列表
2. 有序列表
3. 有序列表

表格

```
|表头|表头|表头|
|:--|:--:|--:|
|内容|内容|内容|
|内容|内容|内容|
```

表头	表头	表头
内容	内容	内容
内容	内容	内容

Git

小调查

- 有多少人听说过 Git?
- 有多少人通过命令行使用过 Git?
- 你们知道哪些 Git 命令?

Git 是什么?

- Git 是一个开源的分布式版本控制系统。

版本控制系统 (VCS) 又是什么?

- 版本控制系统管理一个项目中的各种变更，包括源码、文档和其他数据。

还有其他版本控制系统吗?

- 有的，比如 SVN，不过 Git 是目前最流行的版本控制系统
- 2005 年，BitKeeper 收回了 Linux 内核社区的免费使用权
- Linus Torvalds 为了替代 BitKeeper 开发了 Git

Git 的下载

- https://git-scm.com/downloads

一些术语

- Terminal / Command Line：终端，命令行
- CLI：Command Line Interface，命令行接口
- GUI：Graphical User Interface，图形用户界面
- Code Editor：代码编辑器
- Repository：仓库，存储项目代码的地方
- GitHub：全球最大的开源社区，一个 Git 托管服务
  - 类似的还有 GitLab，Gitee 等等
  - 这两年很火的 HuggingFace 也提供了托管服务

注册 GitHub

- https://github.com/

Git 客户端

- 专用的 Git 图形化客户端
  - GitHub Desktop: https://desktop.github.com/
  - GitKraken: https://www.gitkraken.com/
  - SourceTree: https://www.sourcetreeapp.com/
- 现代编译器也通常集成了 Git 功能
  - VSCode
  - JetBrains 系列
- Git 命令行

有时候也需要使用命令行

- 图形化客户端可能并不支持某些操作
  - 目前的专用 Git GUI 已经发展得相当完善
  - 大部分场景下都不会遇到
- 服务器可能无法安装图形化客户端

Git 有多少个命令?

截至现在一共 145 个。

- Porcelain (瓷器)
  - 44 main commands (add, commit, push, pull, ...)
  - 11 manipulators (config, reflog, replace, ...)
  - 17 interrogators (blame, fsck, rerere, ...)
  - 10 interactors (send-email, p4, svn, ...)
- Plumbing (管道)
  - 19 manipulators (apply, commit-tree, ...)
  - 21 interrogators (cat-file, for-each-ref, ...)
  - 5 syncing (fetch-pack, send-pack, ...)
  - 18 internal (check-attr, sh-i18n, ...)

选择合适的目录

cd = change directory

这个目录不宜过长，且路径中请避免出现中文或者空格。

- Windows：`C:\dev` 或者 `D:\dev` (也不要直接在盘根创建项目，最好是建一层目录)
- 其他操作系统：`~/dev`

本地与远端

Git 仓库存在本地 (local) 与远端 (remote)。

在本地创建一个连接到 GitHub 的仓库，有多种方式：

- 如果是已经在 GitHub 中的仓库
  - git clone https://github.com/user/repo.git
- 如果是本地新建的仓库
  - git init
  - 在 GitHub 上创建一个空白仓库
  - git remote add origin https://github.com/user/repo.git
- 使用 GitHub Desktop：左上角选择 Add > Clone Repository / Create New Repository

创建提交

修改 Git 仓库内的文件后：

- git status：查看本地的状态，包括分支和变更信息
  - git status -s
- git add 用于添加文件到暂存
  - git add file1.md
  - git add file1.md file2.md
  - git add *.md
  - git add .
- git commit 创建提交
  - git commit -m title
  - git commit -m title -m message
  - git commit -am title

与远端同步

- git push 推送更新
  - git push origin master
- git fetch 获取远端
  - git fetch origin master
  - git fetch origin
- git pull 拉取更新

查看历史

- git log 查看历史
  - git log --oneline
  - git log --reverse
  - git log -3
  - git log --before=“2020-08-17”
  - git log --author=“Shigma”
  - git log --after=“one week ago”
  - git log --grep=“GUI”
  - git log -S “GUI”
  - git log hash1...hash2
  - git log file.txt
  - git log --pretty=format:”%an committed %H”

对比变更

- git show 查看历史版本
  - git show 921a2ff
  - git show HEAD
  - git show HEAD~2
  - git show HEAD:file.js
- git diff 对比变更
  - git diff --staged
  - git diff HEAD~2 HEAD
  - git diff HEAD~2 HEAD file.txt

创建和切换分支

Git 的每个提交都位于分支上。Git 的默认分支是 master 或者 main。

- git branch 查看或创建分支
  - git branch bugfix
  - git branch -d bugfix
- git checkout 切换分支
  - git checkout dad47ed
  - git checkout bugfix

合并分支

- git merge 合并分支
  - git merge bugfix
  - git merge --no-ff bugfix
  - git merge --squash bugfix
  - git merge --abort

Fork 与 Pull Request

- 默认情况下你无法写入别人的仓库
  - 仓库作者可以指定一系列 collaborators
  - 更多情况下，你需要 fork 以创建你自己的仓库
  - 你可以在自己的 fork 上进行改动
- 改动完成后，可以创建 Pull Request 到原始仓库

Git 保存了所有历史文件

如果 Git 要存储一个项目的 1000 个版本，每两个版本之间的变化不大，那么 Git 应该怎么做?

- A. 存储每个版本的完整文件
- B. 存储第一个版本，以及后续每个版本与前一个版本的差异
- C. 有其他方法