Node.js Code + Learn

COSCon 2019

Agenda

- Node.js Code + Learn 简介
- 项目结构
  - src, lib, deps
  - test, benchmark, tools
- Node.js core development
  - Building, testing, debugging
- Workflow
  - Pull Request, CI, Code Review
  - Backport & releases
- Code + Learn 注意事项

Node.js Code + Learn 简介

- Node.js 社区自发组织的活动
- Node.js Collaborators（对 Node.js core repository 有 commit 权限的人）事先准备一些比较简单的入门任务，手把手指导来参加的人给 Node.js Core 提一个 Pull Request

目标

- 吸引更多的人参与贡献 Node.js
- 让更多的人了解 Node.js Core 的工作流程。
- https://nodejs.org/en/get-involved/code-and-learn/
- https://github.com/nodejs/code-and-learn/issues/97

关于本次 Node.js Code + Learn

- https://github.com/nodejs/code-and-learn/tree/master/archive/shanghai-2019
- 事先有环境和下载好代码的同学可以询问工作人员连接有限的热点
- 没有环境的同学推荐加群，在有网络的时候做任务，在群里答疑

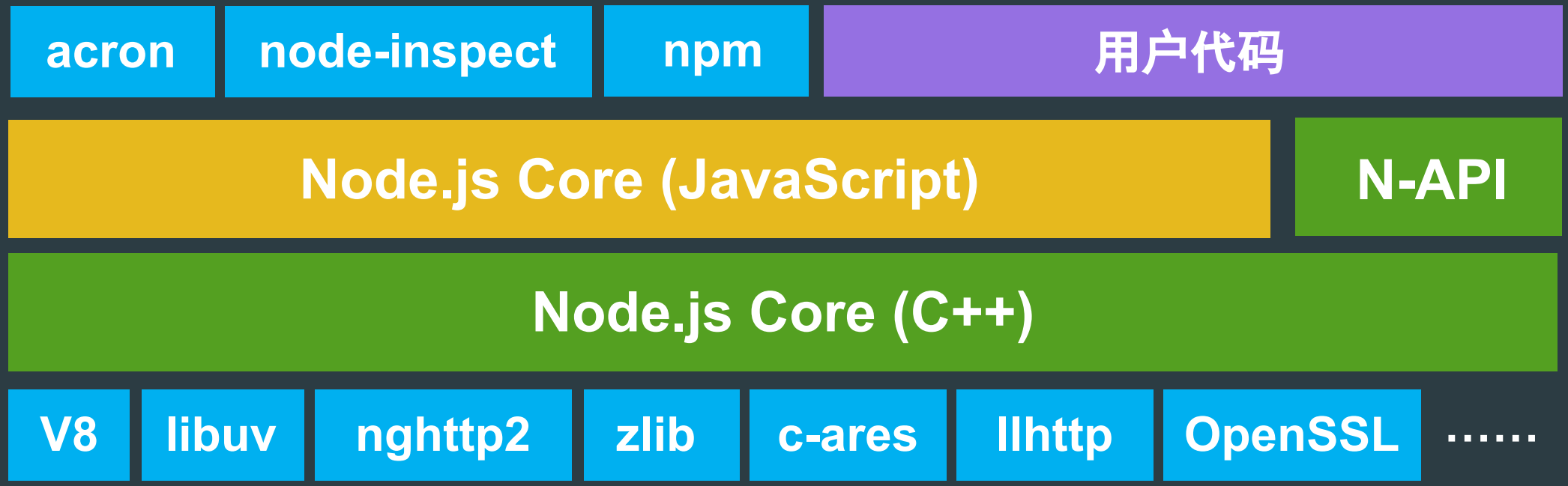
Node.js Code + Learn @ COSCon 19

Valid until 11/10 and will update upon joining group

项目结构 (2019.11)

- src —— C++ 源代码
- lib —— JavaScript 源代码
- deps —— 第三方依赖源代码，使用脚本维护，不用 git submodules
  - v8, openssl, uv, llhttp, npm...
- test —— 测试
- benchmark —— 性能测试（？）
- doc —— 文档（API/少量开发文档）
- tools —— 各种脚本

运行时结构 (2019.11)



C++ 源代码：src/

- node.h ：暴露给 Node.js C++ addon 和 embedders 的头文件
- util.h，util.cc，util-inl.h：公用的一些内部 helpers
- env.h，env.cc，env-inl.h ：Environment 大杂烩类
- 其他大部分以 subsystem 命名，.cc 结尾的文件做实现

JavaScript 源代码：lib/

- lib/internal/ ：内部的 JavaScript 代码，不暴露给用户
- lib/ ：直接放在下面的的文件都暴露给用户可以 require()
- 和 C++ 之间主要通过 process.binding()（过时）和 internalBinding()（不暴露给用户）加载 C++ binding 通信

依赖：deps/

原则上，所有改动都要合并到对应上游项目自己的 repo，才能更新回来。

用脚本负责更新和保留某些特定的补丁（float patches），不用 git submodules

- V8
  - deps/v8/include/v8.h
  - 源代码 https://chromium.googlesource.com/v8/v8.git
  - Change List = Pull Request: https://chromium-review.googlesource.com
  - Issues https://bugs.chromium.org/p/v8
  - 文档 https://v8.dev/docs
- libuv: https://github.com/libuv/libuv
  - deps/uv/include/uv.h

依赖：deps/

- npm：https://github.com/npm/cli
  - 属于 npm 公司，Node.js Core 主要负责复制粘贴
  - 代码就是一个 JavaScript 写的 Node.js 命令行工具
- DNS: c-ares + uv_getaddrinfo
- HTTP parser:
  - http_parser: https://github.com/nodejs/http_parser
  - llhttp https://github.com/nodejs/llhttp
    - (TypeScript DSL-> C)
- Compression
  - zlib
  - brotli
- Crypto: OpenSSL

测试：test/

- parallel：最多测试的地方，可以被并行执行（取决于 CPU 核心数）
- sequential：不能被并行执行的测试，比如要求独占某类系统资源
- cctest：用 gtest 框架写的 C++ 测试
- message：错误信息的测试，用来比对某些 JavaScript 报错内容的修改
- addons：注意有部分测试是从文档 doc/api/addons.md 生成的
- internet：需要网络才能跑的测试，默认不跑（注意GFW）
- known_issues：如名
- pummel：需要跑比较久的测试，默认不跑
- wpt (Web Platform Tests)：Web API 的测试，和浏览器共享，https://github.com/web-platform-tests/wpt 的子集

测试：test/

- doc/guides/writing-tests.md
- tools/test.py: 基于一个老的 V8 的 test runner 改出来的脚本

性能基准测试：benchmark/

- doc/guides/writing-and-running-benchmarks.md
- 用于分析和对比源代码改动带来的性能影响
- node benchmark/compare.js ... > stats.csv
- cat stats.csv | Rscript R benchmark/compare.R ：统计分析性能影响

工具:tools/

- linter (eslint, cpplint.py, remark-lint) & formatter (eslint, clang-format)
  - 格式化 JavaScript 代码: make lint-js-fix
  - 格式化 C++ 代码: CLANG_FORMAT_START=master make format-cpp
- tools/doc: 生成文档 https://nodejs.org/api/
- 各种操作系统的安装包脚本
- 依赖更新脚本
- 生成部分代码
  - v8 inspector, v8 code cache, v8 snapshot
  - js2c.py

文档：doc/

- doc/api: Markdown，经由 tools/doc 下的脚本生成静态 HTML，最终在发布时更新到 https://nodejs.org/api/
- doc/node.1：man page
- doc/guides：给 Node.js Core 开发者看的内部文档
- 官网源代码：https://github.com/nodejs/nodejs.org/

Build (2019.11)

- doc/guides/maintaining-the-build-files.md
- configure / configure.py：定制参数，config.gypi
- vcbuild.bat (Windows) / Makefile (UNIX)
- node.gyp, node.gypi, common.gypi
- tools/js2c.py: 将 lib 下的 JavaScript 代码转换成 const uint8_t[] / const uint16_t[] 编译进二进制文件，就不需要从硬盘读取源代码
- tools/code_cache:生成 JavaScript 代码的 V8 code cache,转换成 const uint8_t[]
- tools/snapshot:生成 V8 Isolate 和 Context 的 snapshot ,转换成 const char[]
- tools/inspector_protocol + src/inspector：生成 V8 inspector 集成用的代码

Build (2019.11)

- libnode:静态/动态库
- node / node.exe (Windows)

Building

MacOS

https://github.com/nodejs/node/blob/master/BUILDING.md#macos-prerequisites

- 从 App store 安装 Xcode
- xcode-select --install

推荐使用 ninja

- brew install ninja

Ubuntu

https://github.com/nodejs/node/blob/master/BUILDING.md#unix-prerequisites

- sudo apt-get update
- sudo apt-get install python2 g++ make

推荐使用 ninja

- sudo apt-get install ninja-build

Building

- python2 ./configure
- make -j4 (4 可替换成 cpu 核心数）

使用 ninja

- python2 ./configure --ninja
- BUILD_WITH=ninja make -j4 (4 可替换成 cpu 核心数）

Building

更多信息参考 BUILDING.md

Testing

- make test-only
- BUILD_WITH=ninja make test-only

运行全部测试（包括文档测试和 linter）

- make test
- BUILD_WITH=ninja make test

Testing

- 匹配：tools/test.py parallel/test-inspector-*
- 运行单个测试： tools/test.py parallel/test-assert
- 约等于（会加 flag，timeout...）：./node parallel/test-assert.js

测试覆盖率

- make coverage
- https://coverage.nodejs.org/

Debugging

JavaScript

- ./node --inspect-brk test.js
- ./node --inspect-brk-node

C++

- ./configure –debug
- lldb -- out/Debug/node test.js
- -O0: ./configure --debug --v8-non-optimized-debug

Node.js Core 简介

- 发行版 release: https://nodejs.org/download/release/
- 源代码，issues & pull requests: https://github.com/nodejs/node
- Collaborators: 对 nodejs/node 有写权限的人，@nodejs/collaborators https://github.com/nodejs/node#collaborators
- TSC (Technical Steering Committee) : Collaborators 的子集，@nodejs/tsc
- 团队组成，加入与退出的流程 :https://github.com/nodejs/node/blob/master/GOVERNANCE.md

Pull Requests

1. Clone & Build
   - git clone git@github.com:nodejs/node.git
2. Hack
   - git checkout –b <branch-name>
   - python2 ./configure --ninja && BUILD_WITH=ninja make test
3. Commit
   - git config user.email <github_account_email>
   - git config user.name <your name>
   - subsystem: title
   - 每行长度 < 72 字符

Pull Requests

- PR
  - fork
  - git remote add <name> git@github.com:<name>/node.git
  - git push –u <name>
  - https://github.com/<name>/node/pull/new/<branch-name>

CI

Travis: 为不是 Collaborators 的人触发，可以提前看到简单的测试结果

Jenkins: https://ci.nodejs.org/job/node-test-pull-request/

- 覆盖 Node.js 支持的大部分平台（操作系统 x CPU 架构 x 编译器 x 定制版本）
- 需要 Collaborator 才有权限手动触发（因为可以在用来编译发布的机器上执行任意代码）
- 机器由 Build Working Group 负责运维 https://github.com/nodejs/build
  - Build WG 管理 SSH/Windows RDP 权限，可以提供临时权限给贡献者用于调试
- GitHub bot 负责更新 Jenkins 状态到 GitHub: https://github.com/nodejs/github-bot

Code Review

- GitHub 上完成
- 需要至少 1 个 Collaborator 的 approval（绿色）
- 1 个 approval 等 7 * 24 小时后才可以合并，2 个 approval 等 2 * 24 个小时
- 根据 review 更新 PR：git commit –fixup / git commit –squash
  - 最后合并时可以配合 git rebase master –i --autosquash
- 没有人负责照看所有 PR，加上每天都会有很多（300+）issues 和 PR 在活动，大部分人不会全部都看
- 如果超过几天没动静需要主动在回复里 ping 一下，直接可以 @ 人，许多人用邮件过滤他们自己的 @ 优先处理

Code Review

黑话 https://chromium.googlesource.com/chromiumos/docs/+/master/glossary.md

LGTM % nits:
S/variable_name/variableName/

我看了觉得没问题（Look Good To Me）
就是有一点小毛病（我只是在挑个刺别往心里去）
麻烦把 variable_name 改成 variableName

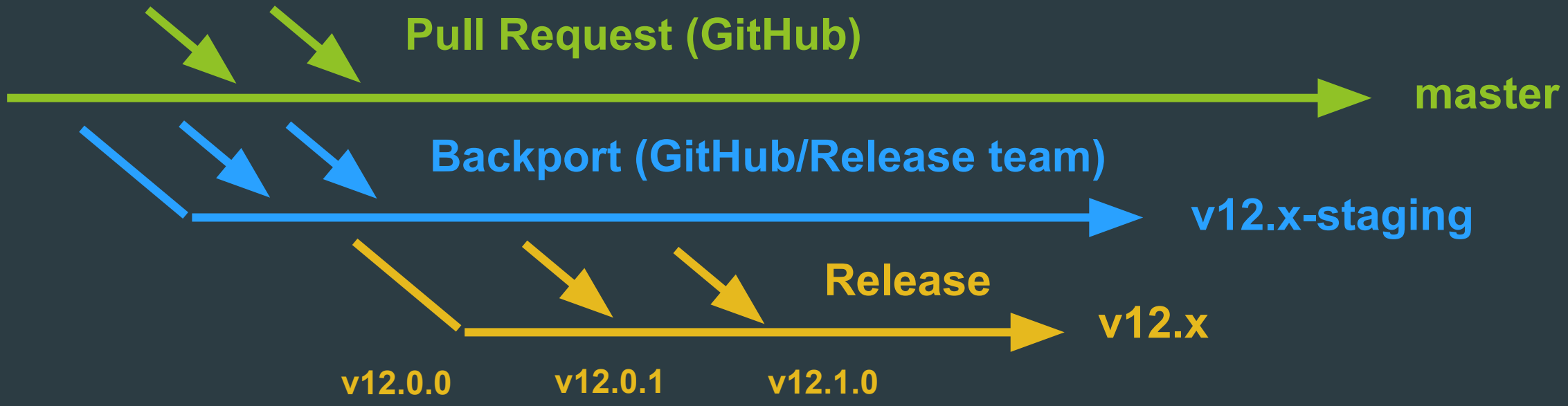
```
$ sed -i -e s/variable_name/variableName/g <file>
$ git add <file>
$ git commit --fixup HEAD <file>
$ git push
```

Fixed, thanks!

合并 Land

- https://github.com/nodejs/node-core-utils
- 定制的 git 命令
  - 检查等待时间和 approval 是否满足要求
  - 有 fast-track 标签的 PR 可以不用等 7/2 天，一般用于紧急情况，比如 master 有测试挂了需要赶紧修复
  - 合并的时候，会给 commit message 加上 PR-URL 和 Review-By 信息，便于日后追踪问题和考古
  - Squash：一个 PR 不是特别复杂，而且专注在一个范围内的改动的话，会全部 squash 成 1 个 commit
  - 每个 commit 必须要能单独跑过测试
  - 检查 commit message 是否符合规范

Backport


Pull Request (GitHub)
master
Backport (GitHub/Release team)
v12.x-staging
Release
v12.x
v12.0.0
v12.0.1
v12.1.0

Release

- @nodejs/release 团队负责管理
- doc/releases.md
- https://ci-release.nodejs.org/
- https://nodejs.org/en/download/

需要贡献的地方

- Automation
  - https://github.com/nodejs/automation
- Build
  - https://github.com/nodejs/build
- Strategic initiatives
  - https://github.com/nodejs/TSC/blob/master/Strategic-Initiatives.md
- Experimental features
- Backport & release
  - https://github.com/nodejs/Release

Code + Learn 注意事项

- https://github.com/nodejs/code-and-learn/tree/master/archive/shanghai-2019
- 不要在一个 PR 里批量修改同类模式的多个文件
  - 如果 a.js 和 b.js 都有 for (var ...) 而你打算修改为 for (let …)，只在一个 PR 里修改 a.js 就好，把 b.js 留给其他人
- 完成一个任务后，切换到另一个任务，不要重复做同一种任务
- https://www.nodetodo.org/next-steps/