Onboarding

This document is an outline of the things we tell new collaborators at their onboarding session.

One week before the onboarding session

- If the new Collaborator is not yet a member of the nodejs GitHub organization, confirm that they are using two-factor authentication. It will not be possible to add them to the organization if they are not using two-factor authentication. If they cannot receive SMS messages from GitHub, try using a TOTP mobile app.
- Announce the accepted nomination in a TSC meeting and in the TSC mailing list.
- Suggest the new Collaborator install `node-core-utils` and set up the credentials for it.

Fifteen minutes before the onboarding session

- Prior to the onboarding session, add the new Collaborator to the collaborators team.
- Ask them if they want to join any subsystem teams. See Who to CC in the issue tracker.

Onboarding session

- This session will cover:
  - local setup
  - project goals and values
  - managing the issue tracker
  - reviewing pull requests
  - landing pull requests

Local setup

- git:
  - Make sure you have whitespace=fix: `git config --global --add apply.whitespace fix`
  - Always create a branch in your own GitHub fork for pull requests
    - Branches in the `nodejs/node` repository are only for release lines
  - Add the canonical nodejs repository as `upstream` remote:
    - `git remote add upstream git://github.com/nodejs/node.git`
  - To update from `upstream`:
    - `git checkout master`
    - `git remote update -p` OR `git fetch --all`
    - `git merge --ff-only upstream/master` (or `REMOTENAME/BRANCH`)
  - Make a new branch for each pull request you submit.
  - Membership: Consider making your membership in the Node.js GitHub organization public. This makes it easier to identify collaborators. Instructions on how to do that are available at Publicizing or hiding organization membership.
- Notifications:
  - Use https://github.com/notifications or set up email
  - Watching the main repository will flood your inbox (several hundred notifications on typical weekdays), so be prepared

The project has a venue for real-time discussion:

- `#nodejs-dev` on the OpenJS Foundation Slack

Project goals and values

- Collaborators are the collective owners of the project
  - The project has the goals of its contributors
- There are some higher-level goals and values
  - Empathy towards users matters (this is in part why we onboard people)
  - Generally: try to be nice to people!
  - The best outcome is for people who come to our issue tracker to feel like they can come back again.
- You are expected to follow *and* hold others accountable to the Code of Conduct.

Managing the issue tracker

- You have (mostly) free rein; don't hesitate to close an issue if you are confident that it should be closed.
  - Be nice about closing issues! Let people know why, and that issues and pull requests can be reopened if necessary.
- See Labels.
  - There is a bot that applies subsystem labels (for example, `doc`, `test`, `assert`, or `buffer`) so that we know what parts of the code base the pull request modifies. It is not perfect, of course. Feel free to apply relevant labels and remove irrelevant labels from pull requests and issues.
  - `semver-{minor,major}`:
    - If a change has the remote *chance* of breaking something, use the `semver-major` label
    - When adding a `semver-*` label, add a comment explaining why you're adding it. Do it right away so you don't forget!
  - Please add the `author-ready` label for pull requests, if applicable.
- See Who to CC in the issue tracker.
  - This will come more naturally over time
  - For many of the teams listed there, you can ask to be added if you are interested
    - Some are WGs with some process around adding people, others are only there for notifications
- When a discussion gets heated, you can request that other collaborators keep an eye on it by opening an issue at the private nodejs/moderation repository.
  - This is a repository to which all members of the `nodejs` GitHub organization (not just collaborators on Node.js core) have access. Its contents should not be shared externally.
  - You can find the full moderation policy here.

Reviewing pull requests

- The primary goal is for the codebase to improve.
- Secondary (but not far off) is for the person submitting code to succeed. A pull request from a new contributor is an opportunity to grow the community.

- Review a bit at a time. Do not overwhelm new contributors.
    - It is tempting to micro-optimize. Don't succumb to that temptation. We change V8 often. Techniques that provide improved performance today may be unnecessary in the future.
- Be aware: Your opinion carries a lot of weight!
- Nits (requests for small changes that are not essential) are fine, but try to avoid stalling the pull request.
    - Identify them as nits when you comment: `Nit: change foo() to bar().`
    - If they are stalling the pull request, fix them yourself on merge.
- Insofar as possible, issues should be identified by tools rather than human reviewers. If you are leaving comments about issues that could be identified by tools but are not, consider implementing the necessary tooling.
- Minimum wait for comments time
    - There is a minimum waiting time which we try to respect for non-trivial changes so that people who may have important input in such a distributed project are able to respond.
    - For non-trivial changes, leave the pull request open for at least 48 hours.
    - If a pull request is abandoned, check if they'd mind if you took it over (especially if it just has nits left).
- Approving a change
    - Collaborators indicate that they have reviewed and approve of the changes in a pull request using GitHub's approval interface
    - Some people like to comment `LGTM` ("Looks Good To Me")
    - You have the authority to approve any other collaborator's work.
    - You cannot approve your own pull requests.
    - When explicitly using `Changes requested`, show empathy – comments will usually be addressed even if you don't use it.
        - If you do, it is nice if you are available later to check whether your comments have been addressed
        - If you see that the requested changes have been made, you can clear another collaborator's `Changes requested` review.
        - Use `Changes requested` to indicate that you are considering some of your comments to block the pull request from landing.
- What belongs in Node.js:
    - Opinions vary – it's good to have a broad collaborator base for that reason!
    - If Node.js itself needs it (due to historical reasons), then it belongs in Node.js.
        - That is to say, `url` is there because of `http`, `freelist` is there because of `http`, etc.
    - Things that cannot be done outside of core, or only with significant pain such as `async_hooks`.
- Continuous Integration (CI) Testing:
    - https://ci.nodejs.org/
        - It is not automatically run. You need to start it manually.
    - Log in on CI is integrated with GitHub. Try to log in now!
    - You will be using `node-test-pull-request` most of the time. Go there now!

        - Consider bookmarking it: https://ci.nodejs.org/job/node-test-pull-request/
    - To get to the form to start a job, click on `Build with Parameters`. (If you don't see it, that probably means you are not logged in!) Click it now!
    - To start CI testing from this screen, you need to fill in two elements on the form:
        - The `CERTIFY_SAFE` box should be checked. By checking it, you are indicating that you have reviewed the code you are about to test and you are confident that it does not contain any malicious code. (We don't want people hijacking our CI hosts to attack other hosts on the internet, for example!)
        - The `PR_ID` box should be filled in with the number identifying the pull request containing the code you wish to test. For example, if the URL for the pull request is `https://github.com/nodejs/node/issues/7006`, then put `7006` in the `PR_ID`.
        - The remaining elements on the form are typically unchanged.
    - If you need help with something CI-related:
        - Use the Build WG repository to file issues for the Build WG members who maintain the CI infrastructure.

Landing pull requests

See the Collaborator Guide: Landing pull requests.

Commits in one pull request that belong to one logical change should be squashed. It is rarely the case in onboarding exercises, so this needs to be pointed out separately during the onboarding.

Exercise: Make a pull request adding yourself to the README

- Example: https://github.com/nodejs/node/commit/b58fe52692659c0bc25ddbe6afa7f4ae2c7f14a8
    - For raw commit message: `git show --format=%B b58fe52692659c0bc25ddbe6afa7f4ae2c7f14a8`
- Collaborators are in alphabetical order by GitHub username.
- Optionally, include your personal pronouns.
- Add the `Fixes: <collaborator-nomination-issue-url>` to the commit message so that when the commit lands, the nomination issue url will be automatically closed.
- Label your pull request with the `doc`, `notable-change`, and `fast-track` labels. The `fast-track` label should cause the Node.js GitHub bot to post a comment in the pull request asking collaborators to approve the pull request by leaving a 👍 reaction on the comment.
- Optional: Run CI on the pull request. Use the `node-test-pull-request` CI task. As a convenience, you may apply the `request-ci` label to the pull request to have a GitHub Actions workflow start the Jenkins CI task for you.
- After two Collaborator approvals for the change and two Collaborator approvals for fast-tracking, land the PR.
- If there are not enough approvals within a reasonable time, consider the single approval of the onboarding TSC member sufficient, and land the pull request.
    - Be sure to add the `PR-URL: <full-pr-url>` and appropriate `Reviewed-By:` metadata.
    - `node-core-utils` automates the generation of metadata and the landing process. See the documentation of `git-node`.
    - `core-validate-commit` automates the validation of commit messages. This will be run during `git node land --final` of the `git-node` command.

Final notes

- Don't worry about making mistakes: everybody makes them, there's a lot to internalize and that takes time (and we recognize that!)
- Almost any mistake you could make can be fixed or reverted.
- The existing collaborators trust you and are grateful for your help!
- Other repositories:
  - https://github.com/nodejs/TSC
  - https://github.com/nodejs/build
  - https://github.com/nodejs/nodejs.org
  - https://github.com/nodejs/readable-stream
  - https://github.com/nodejs/LTS
  - https://github.com/nodejs/citgm
- The OpenJS Foundation hosts regular summits for active contributors to the Node.js project, where we have face-to-face discussions about our work on the project. The Foundation has travel funds to cover participants' expenses including accommodations, transportation, visa fees, etc. if needed. Check out the summit repository for details.
- If you are interested in helping to fix coverity reports consider requesting access to the projects coverity project as outlined in static-analysis.