Maintaining V8 in Node.js

Background

V8 follows the Chromium release schedule. The support horizon for Chromium is different compared to the support horizon for Node.js. As a result, Node.js needs to support multiple versions of V8 longer than what upstream needs to support. V8 branches in Node.js lack of an official maintenance process due to a missing LTS supported branch.

This document attempts to outline the current maintenance processes, proposes a workflow for maintaining the V8 branches in both Node.js LTS and current releases, and discusses how the Node.js and V8 teams at Google can help.

V8 release schedule

V8 and Chromium follow a roughly 6-week release cadence. At any given time there are three V8 branches that are **active**.

For example, at the time of this writing:

- **Stable**: V8 5.4 is currently shipping as part of Chromium stable. This branch was created approx. 6 weeks before from when V8 5.3 shipped as stable.
- **Beta**: V8 5.5 is currently in beta. It will be promoted to stable next; approximately 6 weeks after V8 5.4 shipped as stable.
- **Master**: V8 tip-of-tree corresponds to V8 5.6. This branch gets regularly released as part of the Chromium **canary** builds. This branch will be promoted to beta next when V8 5.5 ships as stable.

All older branches are abandoned and are not maintained by the V8 team.

V8 merge process overview

The process for backporting bug fixes to active branches is officially documented on the V8 wiki. The summary of the process is:

- V8 only supports active branches. There is no testing done on any branches older than the current stable/beta/master.
- A fix needing backport is tagged w/ *merge-request-x.x* tag. This can be done by anyone interested in getting the fix backported. Issues with this tag are reviewed by the V8 team regularly as candidates for backporting.
- Fixes need some 'baking time' before they can be approved for backporting. This means waiting a few days to ensure that no issues are detected on the canary/beta builds.
- Once ready, the issue is tagged w/ *merge-approved-x.x* and one can do the actual merge by using the scripts on the wiki page.
- Merge requests to an abandoned branch will be rejected.
- Only bug fixes are accepted for backporting.

Node.js support requirements

At any given time Node.js needs to be maintaining a few different V8 branches for the various Current, LTS, and nightly releases. At present this list includes the following branches:[1]

Release

Support Start

Support End

V8 version

V8 branch released

V8 branch abandoned

Node.js 4.x

2015-10-01

April 2018

4.5

2015-09-01

2015-10-13

Node.js 6.x

2016-04-01

April 2019

5.1

2016-05-31

2016-06-26

Node.js 8.x

2017-05-30

December 2019

6.1 (soon to be 6.2)

2017-10-17 (6.2)

~2017-12-05 (6.2)

Node.js 9.x

2017-10-31

[1] Node.js 0.12 and older are intentionally omitted from this document as their support has ended.

April 2018

6.2

2017-10-17

~2017-12-05

master

N/A

N/A

6.2

2017-10-17

~2017-12-05

The versions of V8 used in Node.js v4.x, v6.x, and 8.x have already been abandoned by upstream V8. However, Node.js needs to continue supporting these branches for many months (Current branches) or several years (LTS branches).

Maintenance process

Once a bug in Node.js has been identified to be caused by V8, the first step is to identify the versions of Node.js and V8 affected. The bug may be present in multiple different locations, each of which follows a slightly different process.

- Unfixed bugs. The bug exists in the V8 master branch.
- Fixed, but needs backport. The bug may need porting to one or more branches.
  - Backporting to active branches.
  - Backporting to abandoned branches.
- Backports identified by the V8 team. Bugs identified by upstream V8 that we haven't encountered in Node.js yet.

Unfixed upstream bugs

If the bug can be reproduced on the Node.js `canary` branch, Chromium canary, or V8 tip-of-tree, and the test case is valid, then the bug needs to be fixed upstream first.

- Start by opening a bug upstream using this template.
- Make sure to include a link to the corresponding Node.js issue (if one exists).
- If the fix is simple enough, you may fix it yourself; contributions are welcome.
- V8's build waterfall tests your change.

- Once the bug is fixed it may still need backporting, if it exists in other V8 branches that are still active or are branches that Node.js cares about. Follow the process for backporting below.

Backporting to active branches

If the bug exists in any of the active V8 branches, we may need to get the fix backported. At any given time, there are two active branches (beta and stable) in addition to master. The following steps are needed to backport the fix:

- Identify which version of V8 the bug was fixed in.
- Identify if any active V8 branches still contain the bug:
- A tracking bug is needed to request a backport.
  - If there isn't already a V8 bug tracking the fix, open a new merge request bug using this Node.js specific template.
  - If a bug already exists
    - Add a reference to the GitHub issue.
    - Attach *merge-request-x.x* labels to the bug for any active branches that still contain the bug.
- Once the merge has been approved, it should be merged using the merge script documented in the V8 wiki. Merging requires commit access to the V8 repository. If you don't have commit access you can indicate someone on the V8 team can do the merge for you.
- It is possible that the merge request may not get approved, for example if it is considered to be a feature or otherwise too risky for V8 stable. In such cases we float the patch on the Node.js side. See the process on 'Backporting to Abandoned branches'.
- Once the fix has been merged upstream, it can be picked up during an update of the V8 branch (see below).

Backporting to abandoned branches

Abandoned V8 branches are supported in the Node.js repository. The fix needs to be cherry-picked in the Node.js repository and V8-CI must test the change.

- For each abandoned V8 branch corresponding to an LTS branch that is affected by the bug:
  - Checkout a branch off the appropriate *vY.x-staging* branch (e.g. *v6.x-staging* to fix an issue in V8 5.1).
  - Cherry-pick the commit(s) from the V8 repository.
  - Increase the `v8_embedder_string` number in `common.gypi`.
  - In some cases the patch may require extra effort to merge in case V8 has changed substantially. For important issues, we may be able to lean on the V8 team to get help with reimplementing the patch.
  - Open a cherry-pick pull request on `nodejs/node` targeting the *vY.x-staging* branch and notify the `@nodejs/v8` team.
  - Run the Node.js V8 CI in addition to the Node.js CI. The CI uses

the `test-v8` target in the `Makefile`, which uses `tools/make-v8.sh` to reconstruct a git tree in the `deps/v8` directory to run V8 tests.[2]

The `git-node` tool can be used to simplify this task. Run `git node v8 backport <sha>` to cherry-pick a commit.

An example for workflow how to cherry-pick consider the bug RegExp show inconsistent result with other browsers. From the bug we can see that it was merged by V8 into 5.2 and 5.3, and not into V8 5.1 (since it was already abandoned). Since Node.js `v6.x` uses V8 5.1, the fix needed to be cherry-picked. To cherry-pick, here's an example workflow:

- Download and apply the commit linked-to in the issue (in this case a51f429):

  ```
  curl -L https://github.com/v8/v8/commit/a51f429.patch | git am -3 --directory=deps/v8
  ```

  If the branches have diverged significantly, this may not apply cleanly. It may help to try to cherry-pick the merge to the oldest branch that was done upstream in V8. In this example, this would be the patch from the merge to 5.2. The hope is that this would be closer to the V8 5.1, and has a better chance of applying cleanly.

- Modify the commit message to match the format we use for V8 backports and replace yourself as the author. `git commit --amend --reset-author`. You may want to add extra description if necessary to indicate the impact of the fix on Node.js. In this case the original issue was descriptive enough. Example:

```
deps: cherry-pick a51f429 from V8 upstream

Original commit message:
  [regexp] Fix case-insensitive matching for one-byte subjects.

  The bug occurs because we do not canonicalize character class ranges
  before adding case equivalents. While adding case equivalents, we abort
  early for one-byte subject strings, assuming that the ranges are sorted.
  Which they are not.

  R=marja@chromium.org
  BUG=v8:5199

  Review-Url: https://codereview.chromium.org/2159683002
  Cr-Commit-Position: refs/heads/master@{#37833}

Refs: https://github.com/v8/v8/commit/a51f429772d1e796744244128c9feeab4c26a854
PR-URL: https://github.com/nodejs/node/pull/7833
```

[2]The V8 tests still require Python 2. To run these tests locally, you can run `PYTHON2 ./configure.py` before running `make test-v8`, in the root of this repository. On macOS, this also requires a full Xcode install, not just the "command line tools" for Xcode.

- Open a PR against the `v6.x-staging` branch in the Node.js repository. Launch the normal and V8 CI using the Node.js CI system. We only needed to backport to `v6.x` as the other LTS branches weren't affected by this bug.

Backports identified by the V8 team

For bugs found through the browser or other channels, the V8 team marks bugs that might be applicable to the abandoned branches in use by Node.js. This is done through manual tagging by the V8 team and through an automated process that tags any fix that gets backported to the stable branch (as it is likely candidate for backporting further).

Such fixes are tagged with the following labels in the V8 issue tracker:

- `NodeJS-Backport-Review` (V8, Chromium): to be reviewed if this is applicable to abandoned branches in use by Node.js. This list if regularly reviewed by the Node.js team at Google to determine applicability to Node.js.
- `NodeJS-Backport-Approved` (V8, Chromium): marks bugs that are deemed relevant to Node.js and should be backported.
- `NodeJS-Backport-Done` (V8, Chromium): Backport for Node.js has been performed already.
- `NodeJS-Backport-Rejected` (V8, Chromium): Backport for Node.js is not desired.

The backlog of issues with such is regularly reviewed by the node-team at Google to shepherd through the backport process. External contributors are welcome to collaborate on the backport process as well. Some of the bugs may be security issues and will not be visible to external collaborators.

Updating V8

Node.js keeps a vendored copy of V8 inside of the deps/ directory. In addition, Node.js may need to float patches that do not exist upstream. This means that some care may need to be taken to update the vendored copy of V8.

V8 builds against the version of ICU supplied by Node.js, see maintaining-icu.md for special considerations. Specifically, a V8 update may necessitate an ICU update.

Minor updates (patch level)

Because there may be floating patches on the version of V8 in Node.js, it is safest to apply the patch level updates as a patch. For example, imagine that upstream V8 is at 5.0.71.47 and Node.js is at 5.0.71.32. It would be best to compute the diff between these tags on the V8 repository, and then apply that

patch on the copy of V8 in Node.js. This should preserve the patches/backports that Node.js may be floating (or else cause a merge conflict).

The rough outline of the process is:

```
# Assuming your fork of Node.js is checked out in $NODE_DIR
# and you want to update the Node.js master branch.
# Find the current (OLD) version in
# $NODE_DIR/deps/v8/include/v8-version.h
cd $NODE_DIR
git checkout master
git merge --ff-only origin/master
git checkout -b V8_NEW_VERSION
curl -L https://github.com/v8/v8/compare/${V8_OLD_VERSION}...${V8_NEW_VERSION}.patch | git a
# You may want to amend the commit message to describe the nature of the update
```

V8 also keeps tags of the form *5.4-lkgr* which point to the *Last Known Good Revision* from the 5.4 branch that can be useful in the update process above.

The `git-node` tool can be used to simplify this task. Run `git node v8 minor` to apply a minor update.

Major updates

We upgrade the version of V8 in Node.js master whenever a V8 release goes stable upstream, that is, whenever a new release of Chrome comes out.

Upgrading major versions would be much harder to do with the patch mechanism above. A better strategy is to

1. Audit the current master branch and look at the patches that have been floated since the last major V8 update.
2. Replace the copy of V8 in Node.js with a fresh checkout of the latest stable V8 branch. Special care must be taken to recursively update the DEPS that V8 has a compile time dependency on (at the moment of this writing, these are only trace_event and gtest_prod.h)
3. Reset the `v8_embedder_string` variable to "-node.0" in `common.gypi`.
4. Refloat (cherry-pick) all the patches from list computed in 1) as necessary. Some of the patches may no longer be necessary.

To audit for floating patches:

```
git log --oneline deps/v8
```

To replace the copy of V8 in Node.js, use the `git-node` tool. For example, if you want to replace the copy of V8 in Node.js with the branch-head for V8 5.1 branch:

```
cd $NODE_DIR
git node v8 major --branch=5.1-lkgr
```

This should be followed up with manual refloating of all relevant patches.

Proposal: Using a fork repository to track upstream V8

The fact that Node.js keeps a vendored, potentially edited copy of V8 in deps/ makes the above processes a bit complicated. An alternative proposal would be to create a fork of V8 at `nodejs/v8` that would be used to maintain the V8 branches. This has several benefits:

- The process to update the version of V8 in Node.js could be automated to track the tips of various V8 branches in `nodejs/v8`.
- It would simplify cherry-picking and porting of fixes between branches as the version bumps in `v8-version.h` would happen as part of this update instead of on every change.
- It would simplify the V8-CI and make it more automatable.
- The history of the V8 branch in `nodejs/v8` becomes purer and it would make it easier to pull in the V8 team for help with reviewing.
- It would make it simpler to setup an automated build that tracks Node.js master + V8 lkgr integration build.

This would require some tooling to:

- A script that would update the V8 in a specific Node.js branch with V8 from upstream (dependent on branch abandoned vs. active).
- We need a script to bump V8 version numbers when a new version of V8 is promoted from `nodejs/v8` to `nodejs/node`.
- Enabled the V8-CI build in Jenkins to build from the `nodejs/v8` fork.