# Contribution Guidelines

## Patch Submission Process

The following guidelines on the submission process are provided to help you be more effective when submitting code to the JerryScript project.

When development is complete, a patch set should be submitted via GitHub pull requests. A review of the patch set will take place. When accepted, the patch set will be integrated into the master branch, verified, and tested. It is then the responsibility of the authoring developer to maintain the code throughout its lifecycle.

Please submit all patches in public by opening a pull request. Patches sent privately to Maintainers and Committers will not be considered. Because the JerryScript Project is an Open Source project, be prepared for feedback and criticism-it happens to everyone-. If asked to rework your code, be persistent and resubmit after making changes.

### 1. Scope the patch

Smaller patches are generally easier to understand and test, so please submit changes in the smallest increments possible, within reason. Smaller patches are less likely to have unintended consequences, and if they do, getting to the root cause is much easier for you and the Maintainers and Committers. Additionally, smaller patches are much more likely to be accepted.

### 2. Ensure all files have a proper license header and copyright notice

Any code that you want to contribute to the project must be licensed under the [Apache License 2.0](LICENSE). Contributions under a different license can not be accepted. Each file should start with the following header:

```c

/\* Copyright JS Foundation and other contributors, http://js.foundation

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

```

Adding copyright notices other than the project-wide notice ("Copyright JS Foundation and other contributors, http://js.foundation") is not permitted. The only exception is adding third-party code which requires copyright notices to be preserved. Adding third-party code to the project generally requires a strong justification.

### 3. Sign your work with the JerryScript [Developer's Certificate of Origin](DCO.md)

The sign-off is a simple line at the end of the commit message of the patch, which certifies that you wrote it or otherwise have the right to pass it on as an Open Source patch. The sign-off is required for a patch to be accepted.

We have the same requirements for using the signed-off-by process as the Linux kernel.

In short, you need to include a signed-off-by tag in every patch.

You should use your real name and email address in the format below:

> JerryScript-DCO-1.0-Signed-off-by: Random J Developer random@developer.example.org

"JerryScript-DCO-1.0-Signed-off-by:" this is a developer's certification that he or she has the right to submit the patch for inclusion into the project. It is an agreement to the JerryScript [Developer's Certificate of Origin](DCO.md). \*\*Code without a proper signoff cannot be merged into the mainline.\*\*

### 4. Open a GitHub [pull request](https://github.com/jerryscript-project/jerryscript/pulls)

You can find instructions about opening a pull request [here](https://help.github.com/articles/creating-a-pull-request).

### 5. What if my patch is rejected?

It happens all the time, for many reasons, and not necessarily because the code is bad. Take the feedback, adapt your code, and try again. Remember, the ultimate goal is to preserve the quality of the code and maintain the focus of the Project through intensive review.

Maintainers and Committers typically have to process a lot of submissions, and the time for any individual response is generally limited. If the reason for rejection is unclear, please ask for more information from the Maintainers and Committers.

If you have a solid technical reason to disagree with feedback and you feel that reason has been overlooked, take the time to thoroughly explain it in your response.

### 6. Code review

Code review can be performed by all the members of the Project (not just Maintainers and Committers). Members can review code changes and share their opinion through comments guided by the following principles:

\* Discuss code; never discuss the code's author

\* Respect and acknowledge contributions, suggestions, and comments

\* Listen and be open to all different opinions

\* Help each other

Changes are submitted via pull requests and only the Maintainers and Committers should approve or reject the pull request (note that only Maintainers can give binding review scores).

Changes should be reviewed in reasonable amount of time. Maintainers and Committers should leave changes open for some time (at least 1 full business day) so others can offer feedback. Review times increase with the complexity of the review.

## Tips on GitHub Pull Requests

\* [Fork](https://guides.github.com/activities/forking) the GitHub repository and clone it locally

\* Connect your local repository to the original upstream repository by adding it as a remote

\* Create a [branch](https://guides.github.com/introduction/flow) for your edits

\* Pull in upstream changes often to stay up-to-date so that when you submit your pull request, merge conflicts will be less likely

For more details, see the GitHub [fork syncing](https://help.github.com/articles/syncing-a-fork) guidelines.

## How to add the DCO line to every single commit automatically

It is easy to forget adding the DCO line to the end of every commit message. Fortunately there is a nice way to do it automatically. Once you've cloned the repository into your local machine, you can add `prepare commit message hook` in `.git/hooks` directory like this:

```

#!/usr/bin/env python

import sys

commit\_msg\_filepath = sys.argv[1]

with open(commit\_msg\_filepath, "r+") as f:

content = f.read()

f.seek(0, 0)

if "Signed-off-by" not in content:

f.write("\n\nJerryScript-DCO-1.0-Signed-off-by: <Your Name> <Your Email>\n%s" % content)

else:

f.write(content)

```

Please refer [Git Hooks](http://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks) for more information.