# Contribution Guide

This page describes how to contribute changes to PhantomJS.

Please do \*\*not\*\* create a pull request without reading this guide first. Failure to do so may result in the \*\*rejection\*\* of the pull request.

## For The Impatients

\*\*Work on a feature branch\*\*.

If your changes need to be modified due to some reviews, it is less clutter to tweak an isolated feature branch and push it again.

\*\*Create a ticket in the issue tracker\*\*.

This serves as a placeholder for important feedback, review, or any future updates.

In the commit message:

\* \*\*Keep the first line < 72 characters\*\*. Write additional paragraphs

if necessary.

\* \*\*Put the link to the issue\*\* (see above). This is important for cross-referencing purposes.

## Communicate

\*Second opinion is always important.\*

\*\*Bug fixing\*\*. If you have a fix for a bug, please attach your patch in the corresponding issue in the [issue tracker](https://github.com/ariya/phantomjs/issues). If there is no entry for the bug yet, then please create a new one. If you are confident working with Git, see the Get Ready section below on how to submit your change.

\*\*Improvement and feature request\*\*. If you have an improvement idea, please send an email to the [mailing list](http://groups.google.com/group/phantomjs) (rather than contacting the developers directly) so that other people can give their insights and opinions. This is also important to avoid duplicate work.

\*\*Task management\*\*. Once the feature idea is agreed upon and translated into concrete actions and tasks, please use the [issue tracker](https://github.com/ariya/phantomjs/issues) to create an issue for each individual task. Further technical discussion about the task and the implementation details should be carried out in the issue tracker.

\*\*Extending with new API\*\*. Whenever you want to introduce a new API, please send an email to the mailing list along with the link to the issue. Consider good API name for the object or function, read the [API Design Principle](http://wiki.qt.io/API\_Design\_Principles) article. It may require few iterations to agree on the final API and hence it is important to engage all interested parties as early as possible.

## Get Ready

For your proposed change, you need to have:

\* \*\*an issue\*\* (in the issue tracker) which describe your bug or feature

\* \*\*a feature branch\*\* in your git fork

Make sure your code change follows the style guide. To format your code, run `./tools/format-code.sh`.

### Refer the Issue

The commit message needs to link to the issue. This cross-reference is [very important](http://ariya.ofilabs.com/2012/01/small-scale-software-craftsmanship.html) for the following reasons.

First, the commit log is frozen and can not be changed. If it contains a mistake or outdated information, the log can not be amended. However, further updates can be still posted to the linked issue, which can be followed from the commit log itself.

Second, it provides a placeholder for code review and other feedback.

An example of a bad commit log:

Fix Mountain Lion

The above log is too short and useless in the long run. A better version (and note the issue link):

Better support for OS X Mountain Lion.

require('system').os.version should give "10.8 (Mountain Lion)".

https://github.com/ariya/phantomjs/issues/10688

### Use Feature Branch

To isolate your change, please avoid working on the master branch. Instead, work on a \*feature branch\* (often also known as \*topic branch\*). You can create a new branch (example here crash-fix) off the master branch by using:

git checkout -b crash-fix master

Refer to your favorite Git tutorial/book for further detailed help.

Some good practices for the feature branch:

\* Give it a meaningful name instead of, e.g. `prevent-zero-divide` instead of just `fix`

\* Make \*granular\* and \*atomic\* commits, e.g. do not mix a typo fix with some major refactoring

\* Keep one branch for one specific issue. If you need to work on other unrelated issues, create another branch.

## Review and Merge

When your branch is ready, send the pull request.

While it is not always the case, often it is necessary to improve parts of your code in the branch. This is the actual review process.

Here is a check list for the review:

\* It does not break the test suite

\* There is no typo

\* The coding style follows the existing one

\* There is a reasonable amount of comment

\* The license header is intact

\* All examples are still working