# Guidelines for contributing to Janet

Thanks for taking time to contribute to Janet!

Please read this document before making contributions.

## Reporting bugs

\* Check past and current issues to see if your problem has been run into before.

\* If you can't find a past issue for your problem, or if the issues has been closed

you should open a new issue. If there is a closed issue that is relevant, make

sure to reference it.

\* As with any project, include a comprehensive description of the problem and instructions

on how to reproduce it. If it is a compiler or language bug, please try to include a minimal

example. This means don't post all 200 lines of code from your project, but spend some time

distilling the problem to just the relevant code.

\* Add the `bug` tag to the issue.

## Contributing Changes

If you want to contribute some code to the project, please submit a pull request and

follow the below guidelines. Not all changes will be merged, and some pull requests

may require changes before being merged.

\* Include a description of the changes.

\* If there are changes to the compiler or the language, please include tests in the test folder.

The test suites are not organized in any particular way now, so simply add your tests

to one of the test suite files (test/suite0.janet, test/suite1.janet, etc.). You can

run tests with `make test`. If you want to add a new test suite, simply add a file to

the test folder and make sure it is run when`make test` is invoked.

\* Be consistent with the style. For C this means follow the indentation and style in

other files (files have MIT license at top, 4 spaces indentation, no trailing

whitespace, cuddled brackets, etc.) Use `make format` to

automatically format your C code with

[astyle](http://astyle.sourceforge.net/astyle.html). You will probably need

to install this, but it can be installed with most package managers.

For janet code, use lisp indentation with 2 spaces. One can use janet.vim to

do this indentation, or approximate as close as possible. There is a janet formatter

in [spork](https://github.com/janet-lang/spork.git) that can be used to format code as well.

## C style

For changes to the VM and Core code, you will probably need to know C. Janet is programmed with

a subset of C99 that works with Microsoft Visual C++. This means most of C99 but with the following

omissions.

\* No `restrict`

\* Certain functions in the standard library are not always available

In practice, this means programming for both MSVC on one hand and everything else on the other.

The code must also build with emscripten, even if some features are not available, although

this is not a priority.

Code should compile warning free and run valgrind clean. I find that these two criteria are some

of the easiest ways to protect against a large number of bugs in an unsafe language like C. To check for

valgrind errors, run `make valtest` and check the output for undefined or flagged behavior.

### Formatting

Use [astyle](http://astyle.sourceforge.net/astyle.html) via `make format` to

ensure a consistent code style for C.

## Janet style

All janet code in the project should be formatted similar to the code in core.janet.

The auto formatting from janet.vim will work well.

## Suggesting Changes

To suggest changes, open an issue on GitHub. Check GitHub for other issues

that may be related to your issue before opening a new suggestion. Suggestions

put forth without code will be considered, but not necessarily implemented in any

timely manner. In short, if you want extra functionality now, then build it.

\* Include a good description of the problem that is being solved

\* Include descriptions of potential solutions if you have some in mind.

\* Add the appropriate tags to the issue. For new features, add the `enhancement` tag.