# How to Contribute

Thank you for inquiring `facil.io`'s contribution guide. It's people like you and me, that are willing to share our efforts, who help make the world of open source development so inspiring and wonderful.

## Guidelines

### General Guidelines

"Facil" comes from the Spanish word "easy", and this is embedded in `facil.io`'s DNA.

`facil.io` contributions should (ideally) be:

\* \*\*Easy to use\*\*:

clear and concise API, with macros that emulate "named arguments" when appropriate.

\* \*\*Easy to maintain\*\*:

\* \*Modular\*: even at the price of performance and even (although less desired) at the price of keeping things DRY.

Developers should be able to simply remove the module from their implementation if they're not using it.

To clarify, a module should have as small a responsibility as possible without requiring non-core modules. This makes the module easier to maintain and minimizes code fragility and code entanglement.

\* \*Succinctly Commented\*: Too much commenting is noise (we can read code), but too little and a future maintainer might not understand why the code was written in the first place.

\* \*\*Easy to port\*\*:

When possible, code should be portable. This is both true in regards to CPU architecture and in regards to OS and environment.

The project currently has the following limitation that might be addressed in the future:

\* The code requires `kqueue` or `epoll` services from the OS, which means Linux / BSD / macOS.

\* The code assumes a Unix environment (file naming etc').

\* Some of the code (namely some HTTP parts) uses unaligned memory access (requiring newer CPUs and possibly introducing undefined behavior).

\* \*\*Easy to compile\*\*:

The code uses GNU `make` and although we have CMake support, neither CMake nor `configure` should be required at any point.

\* \*\*Easy to manage\*\*:

See the License section below. Contributions must relinquish ownership of contributed code, so licensing and copyright can be managed without the need to reach out to every contributer.

### Community Guideline - Play Nice

As a child, I wasn't any good with people (I'm not sure I'm any better now that I'm older)... which is how come I became good with computers and why we have `facil.io` and other open source projects ;-)

However, I promise to do my best to be a respectful communicator and I ask that you do your best as well.

No matter if discussing a PR (where we might find ourselves entering a heated discussion) or answering an issue (where sometime we find ourselves wondering why people think we work for them)... we should all remember that a little compassion and respect goes a long way.

### Style Guide and Guidelines

A few pointers about code styling (pun intended).

\* Use `clang-format` with the `LLVM` style.

\* Initialize all variables during declaration - even if it's redundant.

\* Use `goto` to move code branches to the end of a function's body.

It makes the main body of the function more readable (IMHO) and should help with branch prediction (similar to how `unlikely` might help, but using a different approach)

## A quick run-down

`facil.io` is comprised of the following module "families":

\* The Core:

This module family comprises `facil.io`'s core. Although it can (mostly) be used outside of `facil.io`, none of the modules in this family can be removed.

The module in comprised of two files: `fio.h` and `fio.c`.

The `fio.h` file can be included more then once and includes some core types, such as binary String support, Arrays, Hash Maps, spinlocks, etc' (see documentation).

\* Dynamic Types (`FIOBJ`) with native JSON support.

This soft type system was designed to make some network oriented tasks easier and is therefore used by many of the other modules.

Unlike most modules, this module is only optional if the core is used independently.

\* HTTP / WebSockets:

The `http` folder refers to the inter-connected HTTP/WebSocket extension / module.

Although this module family seems very entangled, I did my best to make it easy to maintain and extend with a minimum of entanglement.

HTTP request and response modules support virtual function tables for future HTTP/2 extensions. The actual request/response implementations might vary between protocol implementation, but their interface should be version agnostic.

Like most modules, it is optional and can be removed from facil.io without any side-effects.

\* Redis:

The redis engine is in it's own folder, both because it's clearly an "add-on" (even though it's a pub/sub add-on) and because it's as optional as it gets.

This is also a good example for my preference for modular design. The RESP parser is a single file library. It can be easily ported to different projects and is totally separate from the network layer.

\* CLI:

The command line interface extension / module is in the folder `cli` and should be considered and optional add-on. Other modules shouldn't rely on it's existence or absence.

This too, much like the Redis module, is a good example of the preferred modular approach.

### Where to start / Roadmap

Before you start working on a feature, I consider opening a PR to edit this CONTRIBUTING file and letting the community know that you took this feature upon yourself.

Add the feature you want to work on to the following list (or assign an existing feature to yourself). This will also allow us to discuss, in the PR's thread, any questions you might have or any expectations that might effect the API or the feature.

Once you have all the information you need to implementing the feature, the discussion can move to the actual feature's PR.

These are the features that have been requested so far. Even if any of them are assigned, feel free to offer your help:

| Feature | assigned | remarks |

|-------------------|--------------------|----------------------------|

| Documentation | ? Help ? | Placed at [`docs/\_SOURCE`](docs/\_SOURCE) |

| Tests | Never enough | run through [`tests.c`](tests/tests.c) but implement in source files. |

| Early Hints HTTP/1.1 | | |

| SSL/TLS | | See [`fio\_tls\_missing.c`](lib/facil/tls/fio\_tls\_missing.c) for example. |

| WebSocket Client | | Missing cookie retention. |

| HTTP Client | | Missing SSL/TLS, cookie retention and auto-redirect(?) |

| HTTP/2 | | |

| HTTP Router | | RESTfuk without RegEx. i.e.: `/users/(:id)` |

| PostgreSQL | | Wrap `libpq.h` for events + pub/sub engine (?) |

| Gossip (?) | | For Pub/Sub engine scaling |

## License

The project requires that all the code is licensed under the MIT license (though that may change).

Please refrain from using or offering code that requires a change to the licensing scheme or that might prevent future updates to the licensing scheme (I'm considering ISC).

I discovered GitHub doesn't offer a default CLA (Copyright and Licensing Agreement), so I adopted the one used by [BearSSL](https://www.bearssl.org/contrib.html), meaning:

\* the resulting code uses the MIT license, listing me (and only me) as the author. You can take credit by stating that the code was written by yourself, but should attribute copyright and authorship to me (Boaz Segev). This is similar to a "work for hire" approach.

\* I will list meaningful contributions in the CHANGELOG and special contributions will be listed in the README and/or here.

This allows me to circumvent any future licensing concerns and prevent contributors from revoking the license attached to their code.

## Notable Contributions

\* @area55git ([Area55](https://github.com/area55git)) contributed the logo under a [Creative Commons Attribution 4.0 International License.](https://creativecommons.org/licenses/by/4.0/).

\* @cdkrot took the time to test some of the demo code using valgrind, detecting a shutdown issue with in core `defer` library and offering a quick fix.

\* @madsheep and @nilclass took the time to expose a very quite issue (#16) that involved a long processing `on\_open` websocket callback and very short network roundtrips, exposing a weakness in the HTTP/1.x logic.

\* @64 took the time to test the pre-released 0.6.0 version and submit [PR #25](https://github.com/boazsegev/facil.io/pull/25), fixing a silent error and some warnings.

\* Florian Weber (@Florianjw) took time to challenge the RiskyHash draft and [exposed a byte ordering error (last 7 byte reading order)](https://www.reddit.com/r/crypto/comments/9kk5gl/break\_my\_ciphercollectionpost/eekxw2f/?context=3).

\* Chris Anderson (@injinj) did amazing work exploring a 128 bit variation and attacking RiskyHash using a variation on a Meet-In-The-Middle attack, written by Hening Makholm (@hmakholm) on his ([SMHasher fork](https://github.com/hmakholm/smhasher)). The RiskyHash dfraft was updated to address this attack.