# Contributing to Compodoc

[commit-message-format]: https://docs.google.com/document/d/1QrDFcIiPjSLDn3EL15IJygNPiHORgU1\_OOAqWjiDU5Y/edit#

[GitHub]: https://github.com/compodoc/compodoc

[js-style-guide]: https://google.github.io/styleguide/jsguide.html

Contributions are welcome and appreciated. As a contributor, here are the guidelines we would like you

to follow:

- [Stack](#stack)

- [Issues and Bugs](#issue)

- [Feature Requests](#feature)

- [Submission Guidelines](#submit)

- [Coding Rules](#rules)

- [Commit Message Guidelines](#commit)

## Stack

Compodoc is written in TypeScript. The compiling process use Rollup.js

## Architecture and operating

See this folder for more details : [DOCS](./docs/README.md)

### Development process

- open one terminal and run inside compodoc folder : `npm run start`

- open another terminal with the source code of the [demo project](https://github.com/compodoc/compodoc-demo-todomvc-angular), and run `node ../compodoc/bin/index-cli.js -p tsconfig.json -a screenshots -n 'TodoMVC Angular documentation' --includes additional-doc --toggleMenuItems "'all'" -s`

### Debugging process

- open one terminal and run inside compodoc folder : `npm run start`

- open another terminal with the source code of the [demo project](https://github.com/compodoc/compodoc-demo-todomvc-angular), and run `node --inspect ../compodoc/bin/index-cli.js -p tsconfig.json -a screenshots -n 'TodoMVC Angular documentation' --includes additional-doc --toggleMenuItems "'all'" -s`

## <a name="issue"></a> Found an Issue?

If you find a bug in the source code or a mistake in the documentation, you can help us by [submitting an issue](#submit-issue) to our [GitHub Repository][github]. Even better, you can [submit a Pull Request](#submit-pr) with a fix.

## <a name="feature"></a> Want a Feature?

You can \*request\* a new feature by [submitting an issue](#submit-issue) to our [GitHub

Repository][github]. If you would like to \*implement\* a new feature, please submit an issue with a proposal for your work first, to be sure that we can use it.

First open an issue and outline your proposal so that it can be discussed. This will also allow us to better coordinate our efforts, prevent duplication of work, and help you to craft the change so that it is successfully accepted into the project.

## <a name="submit"></a> Submission Guidelines

### <a name="submit-issue"></a> Submitting an Issue

Before you submit an issue, please search the issue tracker, maybe an issue for your problem already exists and the discussion might inform you of workarounds readily available.

We want to fix all the issues as soon as possible, but before fixing a bug we need to reproduce and confirm it. In order to reproduce bugs we will systematically ask you to provide a minimal reproduction scenario. Having a reproducible scenario gives us wealth of important information without going back & forth to you with additional questions like:

- version of compodoc, Node.js used

- `.tsconfig.json` configuration

- and most importantly - a use-case that fails

A minimal reproduce scenario using allows us to quickly confirm a bug (or point out coding problem) as well as confirm that we are fixing the right problem.

We will be insisting on a minimal reproduce scenario in order to save maintainers time and ultimately be able to fix more bugs. Interestingly, from our experience users often find coding problems themselves while preparing a minimal repository. We understand that sometimes it might be hard to extract essentials bits of code from a larger code-base but we really need to isolate the problem before we can fix it.

Unfortunately we are not able to investigate / fix bugs without a minimal reproduction, so if we don't hear back from you we are going to close an issue that don't have enough info to be reproduced.

You can file new issues by filling out our [new issue form](https://github.com/compodoc/compodoc/issues/new).

### <a name="submit-pr"></a> Submitting a Pull Request (PR)

Before you submit your Pull Request (PR) consider the following guidelines:

\* I use [Git flow](http://danielkummer.github.io/git-flow-cheatsheet/) for the development

\* Search [GitHub](https://github.com/compodoc/compodoc/pulls) for an open or closed Pull Request

that relates to your submission.

\* Make your changes in a new git branch

```shell

git checkout -b my-fix-branch develop

```

\* Create your patch, \*\*including appropriate test cases\*\*.

\* Follow our [Coding Rules](#rules).

\* Ensure that all tests pass

\* Generate a new project with angular-cli:

```shell

ng new test-pr

```

\* Test that a new project runs correctly:

```shell

cd test-pr

compodoc -p src/tsconfig.json

```

\* Commit your changes using a descriptive commit message that follows our

[commit message conventions](#commit-message-format).

```shell

git commit -a

```

Note: the optional commit `-a` command line option will automatically "add" and "rm" edited files.

\* Push your branch to GitHub:

```shell

git push origin my-fix-branch

```

\* In GitHub, send a pull request to `compodoc/compodoc:develop`.

\* If i suggest changes then

\* Make the required updates.

\* Re-run the tests and retest your sample generated project to ensure tests are still passing.

\* Rebase your branch and force push to your GitHub repository (this will update your Pull Request):

```shell

git rebase master -i

git push -f

```

That's it! Thank you for your contribution!

#### After your pull request is merged

After your pull request is merged, you can safely delete your branch and pull the changes

from the main (upstream) repository:

\* Delete the remote branch on GitHub either through the GitHub web UI or your local shell as follows:

```shell

git push origin --delete my-fix-branch

```

\* Check out the develop branch:

```shell

git checkout develop -f

```

\* Delete the local branch:

```shell

git branch -D my-fix-branch

```

\* Update your develop with the latest upstream version:

```shell

git pull --ff upstream develop

```

### Fork the compodoc project

Go to the [compodoc project](https://github.com/compodoc/compodoc) and click on the "fork" button. You can then clone your own fork of the project, and start working on it.

[Please read the Github forking documentation for more information](https://help.github.com/articles/fork-a-repo)

### Set NPM to use the cloned project

In your cloned `compodoc` project, type `npm link`.

This will do a symbolic link from the global `node\_modules` version to point to this folder, so when we run `compodoc`, you will now use the development version of Compodoc.

For testing, you will want to generate an application with angular-cli, and run compodoc instaled globally.

```shell

compodoc ...

```

## <a name="rules"></a> Coding Rules

To ensure consistency throughout the source code, keep these rules in mind as you are working:

\* All features or bug fixes \*\*must be tested\*\* by one or more specs (unit-tests or e2e-tests).

\* All public API methods \*\*must be documented\*\*. (Details TBC).

\* We follow [Google's JavaScript Style Guide][js-style-guide], but wrap all code at

\*\*100 characters\*\*.

## <a name="commit"></a> Commit Message Guidelines

We have very precise rules over how our git commit messages can be formatted. This leads to \*\*more

readable messages\*\* that are easy to follow when looking through the \*\*project history\*\*. But also,

we use the git commit messages to \*\*generate the Angular CLI change log\*\*.

### Commit Message Format

Each commit message consists of a \*\*header\*\*, a \*\*body\*\* and a \*\*footer\*\*. The header has a special

format that includes a \*\*type\*\*, a \*\*scope\*\* and a \*\*subject\*\*:

```

<type>(<scope>): <subject>

<BLANK LINE>

<body>

<BLANK LINE>

<footer>

```

The \*\*header\*\* is mandatory and the \*\*scope\*\* of the header is optional.

Any line of the commit message cannot be longer 100 characters! This allows the message to be easier

to read on GitHub as well as in various git tools.

### Revert

If the commit reverts a previous commit, it should begin with `revert: `, followed by the header of the reverted commit. In the body it should say: `This reverts commit <hash>.`, where the hash is the SHA of the commit being reverted.

### Type

Must be one of the following:

\* \*\*build\*\*: Changes that affect the build system or external dependencies

\* \*\*ci\*\*: Changes to our CI configuration files and scripts

\* \*\*docs\*\*: Documentation only changes

\* \*\*feat\*\*: A new feature

\* \*\*fix\*\*: A bug fix

\* \*\*perf\*\*: A code change that improves performance

\* \*\*refactor\*\*: A code change that neither fixes a bug nor adds a feature

\* \*\*style\*\*: Changes that do not affect the meaning of the code (white-space, formatting, missing

semi-colons, etc)

\* \*\*test\*\*: Adding missing tests or correcting existing tests

### Scope

The scope should be the name of the npm package affected as perceived by the person reading changelog generated from the commit messages.

### Subject

The subject contains succinct description of the change:

\* use the imperative, present tense: "change" not "changed" nor "changes"

\* don't capitalize first letter

\* no dot (.) at the end

### Body

Just as in the \*\*subject\*\*, use the imperative, present tense: "change" not There are currently a few exceptions to the "use package name" rule:

\* \*\*packaging\*\*: used for changes that change the npm package layout in all of our packages, e.g. public path changes, package.json changes done to all packages, d.ts file/format changes, changes to bundles, etc.

\* \*\*changelog\*\*: used for updating the release notes in CHANGELOG.md

\* none/empty string: useful for `style`, `test` and `refactor` changes that are done across all packages (e.g. `style: add missing semicolons`)

\*\*Breaking Changes\*\* should start with the word `BREAKING CHANGE:` with a space or two newlines. The rest of the commit message is then used for this.

A detailed explanation can be found in this [document][commit-message-format].

## Financial contributions

We also welcome financial contributions in full transparency on our [open collective](https://opencollective.com/compodoc).

Anyone can file an expense. If the expense makes sense for the development of the community, it will be "merged" in the ledger of our open collective by the core contributors and the person who filed the expense will be reimbursed.

## Credits

### Contributors

Thank you to all the people who have already contributed to compodoc!

<a href="graphs/contributors"><img src="https://opencollective.com/compodoc/contributors.svg?width=890" /></a>

### Backers

Thank you to all our backers! [[Become a backer](https://opencollective.com/compodoc#backer)]

<a href="https://opencollective.com/compodoc#backers" target="\_blank"><img src="https://opencollective.com/compodoc/backers.svg?width=890"></a>

### Sponsors

Thank you to all our sponsors! (please ask your company to also support this open source project by [becoming a sponsor](https://opencollective.com/compodoc#sponsor))

<a href="https://opencollective.com/compodoc/sponsor/0/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/0/avatar.svg"></a>

<a href="https://opencollective.com/compodoc/sponsor/1/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/1/avatar.svg"></a>

<a href="https://opencollective.com/compodoc/sponsor/2/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/2/avatar.svg"></a>

<a href="https://opencollective.com/compodoc/sponsor/3/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/3/avatar.svg"></a>

<a href="https://opencollective.com/compodoc/sponsor/4/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/4/avatar.svg"></a>

<a href="https://opencollective.com/compodoc/sponsor/5/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/5/avatar.svg"></a>

<a href="https://opencollective.com/compodoc/sponsor/6/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/6/avatar.svg"></a>

<a href="https://opencollective.com/compodoc/sponsor/7/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/7/avatar.svg"></a>

<a href="https://opencollective.com/compodoc/sponsor/8/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/8/avatar.svg"></a>

<a href="https://opencollective.com/compodoc/sponsor/9/website" target="\_blank"><img src="https://opencollective.com/compodoc/sponsor/9/avatar.svg"></a>