# Contribute

We'd love for you to contribute to our source code and to make angular-gantt better! Here are the guidelines we'd like you to follow:

- [Issues Guidelines](#issue)

- [Coding Rules](#coding)

- [Branching Guidelines](#branching)

- [Pull Request Guidelines](#pull-request)

- [Git Commit Guidelines](#commit)

- [Release Guidelines](#release)

## <a name="issue"></a> Issues Guidelines

Before you submit your issue search the archive, maybe your question was already answered.

If you didn't found anything related in the archive, you can create a new issue providing:

- AngularJS version

- angular-gantt version

- Browser(s) used

- Long description of the issue

- Data used to reproduce, as JSON or javascript.

- Possible workaround

## <a name="coding"></a> Coding Rules

- Please configure your editor with formatting rules defined in [editorconfig](.editorconfig). You may find an

[EditorConfig](http://editorconfig.org/#download) plugin to automatically configure those settings.

- Try to follow convention used in existing sources.

- Use [JSDoc](http://usejsdoc.org/) to comment your code.

## <a name="branching"></a> Branching Guidelines

- Master branch: `master` is the main development branch. Each stable public version is tagged using this branch.

- Release branches: `MAJOR.MINOR.x` are release branches. They contains minor changes and bugfixes after release of first

major version. Only the latest release branch, and it won't contains new features from master branch and next releases.

- Feature branches: They contains code for a features or a bugfix which needs several commits, and will be merged when

it is tested and ready.

## <a name="pull-request"></a> Pull Request Guidelines

- Create and use a branch with a meaning full name, based on the `develop` branch.

- Your pull request must contain only one feature or bugfix.

- Run `yarn dev` to start the development environment.

- Modify source files located in `src/\*\*` and `demo/\*\*`.

- Keep only minimal changes for the feature or the fix.

- For a new feature, write a test by adding of modifying test files located in `test/\*\*`.

- Run tests with `yarn test`, and ensure all tests are passing.

- Update `README.md` if required.

- Build distribution files and demo app by running `yarn build && yarn build-demo`.

- Build docs by running `yarn build-docs` (requires [mkdocs](http://www.mkdocs.org/) installed).

- Create a Pull Request for your feature branch.

## <a name="commit"></a> Git Commit Guidelines (from AngularJS)

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 AngularJS 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>

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.

### Type

Must be one of the following:

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

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

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

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

semi-colons, etc)

\* \*\*refactor\*\*: A code change that neither fixes a bug or adds a feature

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

\* \*\*test\*\*: Adding missing tests

\* \*\*chore\*\*: Changes to the build process or auxiliary tools and libraries such as documentation

generation

### Scope

The scope could be anything specifying place of the commit change. For example `row`,

`task`, `header`, `options`, `css`, `readme`, etc...

### 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 "changed" nor "changes"

The body should include the motivation for the change and contrast this with previous behavior.

###Footer

The footer should contain any information about \*\*Breaking Changes\*\* and is also the place to

reference GitHub issues that this commit \*\*Closes\*\*.

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

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

## <a name="release"></a> Release Guidelines

Release are performed in interactive mode using [release-it](https://github.com/webpro/release-it).

`yarn release -- [patch|minor|major] [--no-write]`

The optional qualifier passed to task can be, as defined by [semver](http://semver.org/):

- `<version>`: Will use the exact given version.

- `patch`: Will increment version to next patch (default).

- `minor`: Will increment version to next minor.

- `major`: Will increment version to next major.

`.release.json` is the `release-it` configuration.

`yarn dist` automates the preparation of release, and can be run manually to check if everything works as expected.

- Build `angular-gantt`.

- Run tests.

- Populate `dist` directory and `demo/dist` with distribution files.

- Populate `docs` directory, containing docs, demo and files located on website.

- Replace `@@version` by version number in `docs` directory and customize website title and meta tags.

- Copy demo application into `docs` directory.

`yarn dist` won't commit or push anything. A dry-run of the release process can also be performed using `--no-write` option.

Make sure both tasks are working properly before running `yarn release` without the `--no-write` option.

`yarn release` will finally publish the release on github (release & tag) and npm (publish).