# Contributing to Scully

We would love for you to contribute to Scully and help make it even better than it is

today! As a contributor, here are the guidelines we would like you to follow:

- [Code of Conduct](#coc)

- [Question or Problem?](#question)

- [Issues and Bugs](#issue)

- [Feature Requests](#feature)

- [Submission Guidelines](#submit)

- [Coding Rules](#rules)

- [Commit Message Guidelines](#commit)

- [Signing the CLA](#signing-the-cla)

## <a name="coc"></a> Code of Conduct

Help us keep Scully open and inclusive. Please read and follow our [Code of Conduct][coc].

## <a name="question"></a> Got a Question or Problem?

Do not open issues for general support questions as we want to keep GitHub issues for bug reports and feature requests. You've got much better chances of getting your question answered on [Stack Overflow][stackoverflow]) where the questions should be tagged with tag `scully`.

Stack Overflow is a much better place to ask questions since:

- there are thousands of people willing to help on Stack Overflow

- questions and answers stay available for public viewing so your question / answer might help someone else

- Stack Overflow's voting system assures that the best answers are prominently visible.

To save your and our time, we will systematically close all issues that are requests for general support and redirect people to Stack Overflow.

If you would like to chat about the question in real-time, you can reach out via our [gitter channel](https://gitter.im/scullyio/community).

## <a name="issue"></a> Found a Bug?

If you find a bug in the source code, you can help us by

[submitting an issue][github-issue] to our [GitHub Repository][github]. Even better, you can

[submit a Pull Request](#submit-pr) with a fix.

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

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

Repository. 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.

Please consider what kind of change it is:

- For a \*\*Major Feature\*\*, 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.

- \*\*Small Features\*\* can be crafted and directly [submitted as a Pull Request](#submit-pr).

## <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. Having a minimal reproducible scenario gives us a wealth of important information without going back & forth to you with additional questions.

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

We will be insisting on a minimal reproduction 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 reproduction. We understand that sometimes it might be hard to extract essential 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 doesn't have enough info to be reproduced.

You can file new issues by selecting from our [new issue templates][github-choose] and filling out the issue template.

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

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

1. Search [GitHub](https://github.com/scullyio/scully/pulls) for an open or closed PR

that relates to your submission. You don't want to duplicate effort.

1. Be sure that an issue describes the problem you're fixing, or documents the design for the feature you'd like to add.

Discussing the design up front helps to ensure that we're ready to accept your work.

1. Fork the scullyio/scully repo.

1. Make your changes in a new git branch:

```shell

git checkout -b my-fix-branch main

```

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

1. Commit your changes using a descriptive commit message that follows our

[commit message conventions](#commit). Adherence to these conventions

is necessary run the command `npm run commit` (this add all the files using `git add .`) or

if you need only add some files, you can need run the command `npm run commit:select`.

1. Push your branch to GitHub:

```shell

git push origin my-fix-branch

```

1. In GitHub, send a pull request to `scully:main`.

- If we suggest changes then:

- Make the required updates.

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

```shell

git rebase main -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 main branch:

```shell

git checkout main -f

```

- Delete the local branch:

```shell

git branch -D my-fix-branch

```

- Update your main with the latest upstream version:

```shell

git pull --ff upstream main

```

## <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\*\*.

### 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\*\*:

\_The command `npm run commit` was previously configured to use all these rules\_

```

<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.

The footer should contain a [closing reference to an issue](https://help.github.com/articles/closing-issues-via-commit-messages/) if any.

Samples from Angular Repository: (even more [samples](https://github.com/angular/angular/commits/main))

```

docs(changelog): update changelog to beta.5

```

```

fix(release): need to depend on latest ng-lib

The version in our package.json gets copied to the one we publish, and users need the latest of these.

```

### Type

Must be one of the following:

- \*\*build\*\*: Changes that affect the build system or external dependencies (example scopes: gulp, broccoli, npm)

- \*\*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 the changelog generated from commit messages).

The following is the list of supported scopes:

- \*\*scully\*\*

- \*\*ng-lib\*\*

- \*\*schematics\*\*

### Subject

The subject contains a succinct description of the change:

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

- don't capitalize the 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\*\*.

\*\*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.

## Signing the CLA

Please sign our Contributor License Agreement (CLA) before sending pull requests. For any code changes to be accepted, the CLA must be signed. It's a quick process, we promise!

- For individuals we have a [simple click-through form][cla-individual].

- For corporations we'll need you to [print, sign and one of scan+email, fax or mail the form][cla-corporations].

[coc]: https://github.com/scullyio/scully/blob/main/CODE\_OF\_CONDUCT.md

[github]: https://github.com/scullyio/scully

[github-issue]: https://github.com/scullyio/scully/issues/new?assignees=&labels=bug&template=---bug-report.md&title=

[github-feature]: https://github.com/scullyio/scully/issues/new?assignees=&labels=enhancement&template=---feature-request.md&title=

[github-choose]: https://github.com/scullyio/scully/issues/new/choose

[stackoverflow]: http://stackoverflow.com/questions/tagged/scully

[cla-individual]: https://cla-assistant.io/scullyio/scully

[cla-corporations]: https://gist.github.com/Villanuevand/4178efad870e8397565c6944fc3d292d