# Contributing to Angular Universal

We would love for you to contribute to Angular 2 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](#cla)

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

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

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

If you have questions about how to \*use\* Angular 2 Universal, please direct them to [Gitter][gitter].

There is also a [Slack](https://angular-universal.slack.com) ground that we would be hapy to invite you to. Just ping [@jeffwhelpley](https://twitter.com/jeffwhelpley)

or [@gdi2290](https://twitter.com/gdi2290) on [Twitter](https://twitter.com) or [Gitter][gitter].

## <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. Angular 2 is in developer preview

and we are not ready to accept major contributions ahead of the full release.

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, search the archive, maybe your question was already answered.

If your issue appears to be a bug, and hasn't been reported, open a new issue.

Help us to maximize the effort we can spend fixing issues and adding new

features, by not reporting duplicate issues. Providing the following information will increase the

chances of your issue being dealt with quickly:

\* \*\*Overview of the Issue\*\* - if an error is being thrown a non-minified stack trace helps

\* \*\*Angular Universal Version\*\* - what version of Angular is affected (e.g. 2.0.0-beta.6)

\* \*\*Motivation for or Use Case\*\* - explain what are you trying to do and why the current behavior is a bug for you

\* \*\*Browsers and Operating System\*\* - is this a problem with all browsers?

\* \*\*Reproduce the Error\*\* - provide a live example (using [Plunker][plunker],

[JSFiddle][jsfiddle] or [Runnable][runnable]) or a unambiguous set of steps

\* \*\*Related Issues\*\* - has a similar issue been reported before?

\* \*\*Suggest a Fix\*\* - if you can't fix the bug yourself, perhaps you can point to what might be

causing the problem (line of code or commit)

You can file new issues by providing the above information [here](https://github.com/angular/universal/issues/new).

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

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

\* Search [GitHub](https://github.com/angular/universal/pulls) for an open or closed PR

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

\* Please sign our [Contributor License Agreement (CLA)](#cla) before sending PRs.

We cannot accept code without this.

\* Make your changes in a new git branch:

```shell

git checkout -b my-fix-branch master

```

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

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

\* Run the full Angular test suite, as described in the [developer documentation][dev-doc],

and ensure that all tests pass.

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

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

is necessary because release notes are automatically generated from these messages.

```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 `universal:master`.

\* If we suggest changes then:

\* Make the required updates.

\* Re-run the Angular 2 test suites for JS and Dart 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 master branch:

```shell

git checkout master -f

```

\* Delete the local branch:

```shell

git branch -D my-fix-branch

```

\* Update your master with the latest upstream version:

```shell

git pull --ff upstream master

```

## <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).

\* 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\*\*. An automated formatter is available, see

[DEVELOPER.md](DEVELOPER.md#clang-format).

## <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 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:

\* \*\*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 nor adds a feature

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

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

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

\* \*\*chore\*\*: Other changes that don't modify `src` or `test` files

### Scope

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

`express-engine`, `common`, 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\*\*.

\*\*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].

## <a name="cla"></a> 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][individual-cla].

\* For corporations we'll need you to

[print, sign and one of scan+email, fax or mail the form][corporate-cla].

[coc]: https://github.com/angular/code-of-conduct/blob/master/CODE\_OF\_CONDUCT.md

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

[corporate-cla]: http://code.google.com/legal/corporate-cla-v1.0.html

[dev-doc]: https://github.com/angular/angular/blob/master/DEVELOPER.md

[github]: https://github.com/angular/universal

[gitter]: https://gitter.im/angular/universal

[individual-cla]: http://code.google.com/legal/individual-cla-v1.0.html

[js-style-guide]: http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml

[jsfiddle]: http://jsfiddle.net/

[plunker]: http://plnkr.co/edit

[runnable]: http://runnable.com/