# New Issues are for bug reports or feature requests only!! Questions on how to use the grid should be posted to Stack Overflow or Gitter https://gitter.im/angular-ui/ui-grid.

# Contributing to UI Grid

Please take a moment to review this document in order to make the contribution

process easy and effective for everyone involved.

Following these guidelines helps to communicate that you respect the time of

the developers managing and developing this open source project. In return,

they should reciprocate that respect in addressing your issue or assessing

patches and features.

## Developer guidelines

If you're a developer, please take a look at the [Developer guidelines](DEVELOPER.md)

## Using the issue tracker

The [issue tracker](https://github.com/angular-ui/ui-grid/issues) is

the preferred channel for [bug reports](#bugs), [features requests](#features)

and [submitting pull requests](#pull-requests), but please respect the following

restrictions:

\* Please \*\*do not\*\* use the issue tracker for personal support requests (use

[Stack Overflow](http://stackoverflow.com/questions/tagged/angular-ui-grid) ( [Old Tag](http://stackoverflow.com/questions/tagged/ng-grid))

or our [Gitter Chat](https://gitter.im/angular-ui/ui-grid)).

\* Please \*\*do not\*\* derail or troll issues. Keep the discussion on topic and

respect the opinions of others.

<a name="bugs"></a>

## Bug reports

A bug is a \_demonstrable problem\_ that is caused by the code in the repository.

Good bug reports are extremely helpful - thank you!

Guidelines for bug reports:

1. \*\*Use the GitHub issue search\*\* &mdash; check if the issue has already been

reported.

2. \*\*Check if the issue has been fixed\*\* &mdash; try to reproduce it using the

latest `master` or development branch in the repository.

3. \*\*Isolate the problem\*\* &mdash; ideally create a [reduced test

case](http://css-tricks.com/6263-reduced-test-cases/) and a live example using [plunker.](http://plnkr.co/edit/bsiESw?p=info)

A good bug report shouldn't leave others needing to chase you up for more

information. Please try to be as detailed as possible in your report. What is

your environment? What steps will reproduce the issue? What browser(s) and OS

experience the problem? What would you expect to be the outcome? All these

details will help people to fix any potential bugs.

Example:

> Short and descriptive example bug report title

>

> A summary of the issue and the browser/OS environment in which it occurs. If

> suitable, include the steps required to reproduce the bug.

>

> 1. This is the first step

> 2. This is the second step

> 3. Further steps, etc.

>

> `<url>` - a link to the reduced test case

>

> Any other information you want to share that is relevant to the issue being

> reported. This might include the lines of code that you have identified as

> causing the bug, and potential solutions (and your opinions on their

> merits).

<a name="features"></a>

## Feature requests

Feature requests are welcome. But take a moment to find out whether your idea

fits with the scope and aims of the project. It's up to \*you\* to make a strong

case to convince the project's developers of the merits of this feature. Please

provide as much detail and context as possible. Please also see the note in the

[Developer guidelines](DEVELOPER.md) about implementing new features as plugins.

<a name="pull-requests"></a>

## Pull requests

Good pull requests - patches, improvements, new features - are a fantastic

help. They should remain focused in scope and avoid containing unrelated

commits.

\*\*Please ask first\*\* before embarking on any significant pull request (e.g.

implementing features, refactoring code, porting to a different language),

otherwise you risk spending a lot of time working on something that the

project's developers might not want to merge into the project.

Please adhere to the coding conventions used throughout a project (indentation,

accurate comments, etc.) and any other requirements (such as test coverage).

\*\*Include Examples\*\* If possible, include links to a youtube video or GIF

demonstrating your fix or your feature. It will help us understand what you

are doing and why we want to merge it in.

Adhering to the following this process is the best way to get your work

included in the project:

1. [Fork](http://help.github.com/fork-a-repo/) the project, clone your fork,

and configure the remotes:

```bash

# Clone your fork of the repo into the current directory

git clone https://github.com/<your-username>/ui-grid.git

# Navigate to the newly cloned directory

cd ui-grid

# Assign the original repo to a remote called "upstream"

git remote add upstream https://github.com/angular-ui/ui-grid.git

```

2. If you cloned a while ago, get the latest changes from upstream:

```bash

git checkout master

git pull upstream master

```

3. Create a new topic branch (off the main project development branch) to

contain your feature, change, or fix:

```bash

git checkout -b <topic-branch-name>

```

4. Commit your changes in logical chunks. Please adhere to these [git commit

message guidelines](http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html)

or your code is unlikely be merged into the main project. Use Git's

[interactive rebase](https://help.github.com/articles/interactive-rebase)

feature to tidy up your commits before making them public.

5. Locally merge (or rebase) the upstream development branch into your topic branch:

```bash

git pull [--rebase] upstream master

```

6. Squash your change into one commit

```bash

git rebase -i master

```

7. Push your topic branch up to your fork:

```bash

git push [--force] origin <topic-branch-name>

```

8. [Open a Pull Request](https://help.github.com/articles/using-pull-requests/)

with a clear title and description.

\*\*IMPORTANT\*\*: By submitting a patch, you agree to allow the project owners to

license your work under the terms of the [MIT License](LICENSE.md).

## <a name="commit"></a> Git Commit 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 UIGrid change log\*\*.

Alternatively you can install [commitizen](https://github.com/commitizen/cz-cli) \*\*OR\*\*

use the wizard. To use the wizard, run `npm run commit` in your terminal after staging your changes in git.

### Commit Message Format

\*\*If you do not follow this style guide your commit message will fail to be created.\*\*

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

\* \*\*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 `core`,

`infinite-scroll`, `pagination`, `cellnav`, `edit`, `importer`, `pinning`, 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.