# Contributing to this project

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.

## Using the issue tracker

The issue tracker 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/mahapps.metro) or [Gitter](https://gitter.im/MahApps/MahApps.Metro)).

\* 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; create an example or perhaps even a test.

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 [the version](#version) of the library which you are using.

- What is your environment?

- What steps will reproduce the issue?

- What OS experience the problem?

- What would you expect to be the outcome?

All these details will help people to fix any potential bugs.

<a name="repro"></a>

## A repro is most excellent

If you can include a working demo of the issue, you're 90% of the way there. This helps us greatly to see the problem and debug it much quicker. Without it, we need to understand how you are using it, which takes time.

<a name="version"></a>

## What version?

Please include the version of the library which you are using. This will help us to isolate the issue quicker.

## Details are crucial

Pictures help to explain visual issues. A video or screencast helps to see animation issues. If these are relevant to your use cases, please include them.

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

## Bugs vs Features

A bug is something that is wrong with a specific version of the application. A feature is a missing or new piece of functionality.

While we are happy to hear about requested features, new development is not a high-priority in the short-term.

<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.

Follow this process if you'd like your work considered for inclusion 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>/<repo-name>

# Navigate to the newly cloned directory

cd <repo-name>

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

git remote add upstream https://github.com/<upstream-owner>/<repo-name>

```

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

```bash

git checkout <dev-branch>

git pull upstream <dev-branch>

```

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 <dev-branch>

```

6. Push your topic branch up to your fork:

```bash

git push origin <topic-branch-name>

```

7. [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 owner to license your work under the same license as that used by the project.

Most text of this contributing is taken from this [issue-guidelines](https://github.com/necolas/issue-guidelines).