# Contributing

We ?? contributions, so if you find a bug or have a great idea for a feature create an [issue](https://github.com/lukasoppermann/html5sortable/issues) or even better, send a [pull request](https://github.com/lukasoppermann/html5sortable/pulls).

We are a beginners friendly project! This means, we are happy to help you with contributing to the project. If you are not sure how to make your PR fulfill all the points below, send a [WIP] (work in progress) PR. In the description you can mentioning the points you are still working on and what you need help with.

If you are just starting out with github, you are very welcome to submit small PRs. For example improvements to the readme or code cleanup PRs.

## Pull Requests

- We use [standardjs](https://standardjs.com/) to enforce common coding style, so please write your code according to it. (Run `npm test` to see where you need adjust your code.)

- \*\*Add tests!\*\* - To avoid bugs we keep the test coverage as high as possible. Please help by adding tests to your bugfixes & features.

- \*\*Comment your code\*\* - Your code should be as self-documenting and easy to read as possible, however please add comments whenever possible/sensible. People with all levels of skill and involvement in the project will work on the code.

- \*\*Use Docblocks for functions\*\* – Every function should have a [docblock](http://usejsdoc.org/about-getting-started.html) above stating what the function does and what parameters it is supposed to receive, return values, etc.

```javascript

/\*

\* remove event handlers from sortable

\* @param: {Element} sortable

\*/

```

- \*\*Document any change in behaviour\*\* - Make sure the `README.md` and any other relevant documentation are kept up-to-date.

- \*\*Do NOT commit the dist folder\*\* – Please do not commit anything in the `dist` directory, those files will be updated once a release is created.

- \*\*One pull request per feature\*\* - We appreciate every improvement you have for the project. To make it easy and fast to merge your pull requests, only change one thing per PR. We rather have 10 small PRs than one giant one.

- \*\*Rebase to master\*\*

Please make sure to [`rebase` to `master`](#rebase-to-master) so that your pull request can be easily merged.

\*\*Remember:\*\* If you have trouble with any of the steps above, just ask for help.

## Running Tests

``` bash

$ npm test

```

## Rebase

If you are new to rebasing you might want to create a backup of your branch, in case something goes wrong ?.

### Rebase to master

#### Track the upstream master

First you need to track the `upstream/master` (master of this repo) from your fork.

```bash

git remote add upstream git@github.com:lukasoppermann/html5sortable.git

```

#### Rebase to the upstream master

Before you send a PR, or whenever you need to get changes from the upstream master into your branch, you need to do the following:

```bash

# get the latest changes from upstream

git fetch upstream

# go to your branch if you are not already on it

git checkout your-branch

# move your changes ontop of the current upstream/master branch

git rebase upstream/master

```

If you are unlucky you might get a conflict. You will need to resolve it just like a merge conflict.

### Interactive rebase

It is nice to try to send only meaningful commits, as this makes it easier for us to understand what you where doing. However when working one often has commits like `fix` or `update`.

To remove those and create an easy to understand history you can use `git rebase -i` – an interactive `-i` rebase.

```bash

# to work on the last 3 commits

git rebase -i HEAD~3

# to work on all commits up to 34kl24314

git rebase -i 34kl24314

```

First you will get an overview of your commits:

```bash

pick ae23f76 update contribution

pick 41c2535 update contribute

```

You can `pick` (keep), `reword` (keep & edit message), `edit`, `squash` (meld into the previous (shown) commit), `fixup` (like squash but without commit message), `exec` or `drop` (remove commit). Those options are listen as comments below the commits in your overview screen. You can also reorder commit to change the order in which they are shown.

\*\*Note:\*\* This screen is read from bottom to top, so for e.g. `squash` will meld into the commit above.

Once you are happy with you changes, save and close the file to move on. Another file with all commit messages is shown. Remove the ones you don't want and save and close this file as well. Your rebase should continue and you should be down.

\*\*Note:\*\* As you rewrote history, you will now need to force-push if you changed commits that are already on a remote server.

\*\*Happy coding!\*\*