# Contributing Guidelines

Some basic conventions for contributing to this project.

### General

Please make sure that there aren't existing pull requests attempting to address the issue mentioned. Likewise, please check for issues related to update, as someone else may be working on the issue in a branch or fork.

\* Non-trivial changes should be discussed in an issue first

\* Develop in a topic branch, not master

\* Squash your commits

### Linting

Please check your code using `npm run lint` before submitting your pull requests, as the CI build will fail if `eslint` fails.

### Commit Message Format

Each commit message should include a \*\*type\*\*, a \*\*scope\*\* and a \*\*subject\*\*:

```

<type>(<scope>): <subject>

```

Lines should not exceed 100 characters. This allows the message to be easier to read on github as well as in various git tools and produces a nice, neat commit log ie:

```

#459 refactor(utils): create url mapper utility function

#463 chore(webpack): update to isomorphic tools v2

#494 fix(babel): correct dependencies and polyfills

#510 feat(app): add react-bootstrap responsive navbar

```

#### 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 or adds a feature

\* \*\*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 `webpack`,

`helpers`, `api` 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