# Contribution documentation

Pull requests and contributions are warmly welcome.

Please follow existing code style and commit message conventions. Also remember to keep documentation

updated.

\*\*Pull requests:\*\* You don't need to bump version numbers or modify anything related to releasing. That stuff is fully automated, just write the functionality.

## Get started with documentation

Documentation is written in markdown and [mkdocs](https://github.com/mkdocs/mkdocs) is used

to build documentation site. [ReadTheDocs](https://readthedocs.org/projects/progressbarjs/)

provides hosting for the documentation site and builds the documentation on each

commit.

To locally develop documentation, you need to install mkdocs and requirements specified

in pip-requirements.txt. After that you can run `mkdocs serve`.

## Get started with development

\* [Install local environment](#install-environment).

\* `cd local-dev` and serve folder to browser

\* Open another terminal to project root and run `grunt watch`.

Then you can edit `src/progressbar.js` and changes can be tested in browser.

Edit `local-dev/main.js` to your testing needs.

Shorter way to do local development is running: ```npm run dev```.

## General project stuff

This package uses npm/node tools just in the developer environment. Grunt is used as a task runner

but there's no reason it couldn't be replaced with e.g. Makefile

\*ProgressBar.js\* depends on tweening library called [shifty](https://github.com/jeremyckahn/shifty).

\*Shifty\* is bundled inside the scripts in [dist/](https://github.com/kimmobrunfeldt/progressbar.js/blob/master/dist) directory.

Dependency is bundled in to ease using the library.

#### Versioning

Versioning follows [Semantic Versioning 2.0.0](http://semver.org/). The release script makes sure

that for each release, there exists only one commit in history where version number in \*bower.json\*

matches the release's version. That commit is tagged as the release, for example `0.4.1`. Commits after that have -dev suffix(\*0.4.1-dev\*) in the version number to avoid any possible confusion.

In other words, if you look into \*bower.json\*, you can tell if the code base is a released version or not.

## {install-environment} Install environment

Install tools needed for development:

npm install

npm install -g watchify

npm install -g browserify

For testing:

npm install -g mocha

npm install -g testem

## Test

Tests are written with [Mocha](http://mochajs.org/) + [expect.js](https://github.com/LearnBoost/expect.js/).

Sometimes the tests fail even though they actually work on the browser. That might be cause of setTimeouts used

in tests.

Quickly run tests:

grunt test

This will use testem to run tests with Chrome. Other options specified below.

#### Testem

[Testem](https://github.com/airportyh/testem) is used for running tests locally. It is fast and easy to use.

List of example commands:

\* `testem` Serves testing page so that you can connect any browser to it.

\* `testem ci` Runs tests an all available/detected local browsers.

\* `testem ci -R dot -l chrome` Runs tests with Chrome using dot reported.

#### Karma

[Karma](http://karma-runner.github.io/) is used for running tests to be executed in Sauce Labs.

Karma was used because integrating it to Sauce Labs is easier than with Testem.

You must setup `SAUCE\_USERNAME` and `SAUCE\_ACCESS\_KEY` environment variables.

See [karma-sauce-launcher project documentation](https://github.com/karma-runner/karma-sauce-launcher#username). Browsers used to test are specified in [karma.conf.js](karma.conf.js).

You can run tests with all browsers:

grunt karma

Tests are run sequentially in batches of browsers to prevent timeouts in Sauce Labs.

You can also run single set of browsers to see test results faster:

grunt karma:sauce0

## Release

\*\*Before releasing, make sure there are no uncommitted files,

tests pass and jshint gives no errors.\*\*

Creating a new release of the package is simple:

1. Commit and push all changes

2. Run local tests and linters with `npm test`

3. Make sure Sauce Labs tests pass

4. Run `grunt release`, which will create new tag and publish code to GitHub

Bower detects your new version of git tag.

5. Edit GitHub release notes

By default, patch release is done. You can specify the version bump as a parameter:

grunt release:major

Valid version bump values: `major`, `minor`, `patch`.

## Decision log

\* Animate SVG paths with CSS3 transitions to make animations smooth.

\* API must provide built-in shapes and a way to use totally custom SVG.

\* Document manually. More overhead and risk of out dated information but easier to get started and contribute with pull requests.

\* Animate paths with JS because IE does not support CSS transitions for SVG properties. This also allows

animation customizations and possible even using different easings per animation(in future).

\* Expose ProgressBar so it can be used with basic module loaders or as a global.

\* Bundle shifty inside the final distributable instead of requiring users to install both libs. If someone has already included shifty, then a custom build should be made.

\* Ship distributables to Bower. Fully automate releasing.

\* Ship distributables also to NPM to ease life of Browserify users.

\* Delegate shifty dependency handling to NPM instead of keeping it in repository but still bundle it inside the final distributable.

\* Automate tests so that testing is locally fast and CI runs tests with more browsers in Sauce Labs

\* Because of introducing text attribute, the library must modify CSS also. Provide user an option to make CSS them selves.

\* Keep \*\*master\*\* branch as a release branch so that new users can see the documentation for latest release instead of development version.

\* Add some tasks to package.json aside with grunt tasks. I would want to move from Grunt to npm scripts totally but Sauce Labs tests are run with grunt.