# Contribution Guide

Thank you for considering a contribution to `pwa-asset-generator`. Before jumping into coding, please go through this guide carefully for a successful contribution.

## Code of conduct

First of all, please read the [\_\_Code of Conduct\_\_](https://github.com/onderceylan/pwa-asset-generator/blob/master/CODE\_OF\_CONDUCT.md) carefully. Once you start contributing, you agree on all terms in the code of conduct without any exception.

## Where to start

You can contribute to this project in many aspects. Some are; reporting bugs, submitting feature requests, reviewing pull requests, improving documentation, adding new test scenarios, implementation of bug fixes, implementation of feature requests and etc. A good issue candidate for you to start with, is an issue with `good first issue` or `help wanted` labels. For all other issues, please contact one of the core contributors before working on them to prevent duplicate work.

Please note that if any of the issues have an assignee or is `In Progress` within projects, it means it's assigned to a specific contributor and it's in progress already. You can see the milestones and road map under [projects](https://github.com/onderceylan/pwa-asset-generator/projects) section.

## Code style

\* This project uses an opinionated code style via [prettier](https://github.com/prettier/prettier).

\* Every time you commit, eslint prettier plugin fixes any formatting issues via git commit hooks to keep the code style consistent across the project. Please do not disable commit hooks while committing.

\* To enable commit message linting via git hooks, please run `npm install` before committing any change.

## Commit messages and continuous deployment

This project uses [husky](https://github.com/typicode/husky) w [commitizen](https://github.com/commitizen/cz-cli) and [semantic-release](https://github.com/semantic-release/semantic-release) for conventional changelog and continuous deployment.

When you run `npm install` before introducing any change, all the necessary packages for this workflow will be installed on your local repo clone.

You should commit your changes without providing a custom message via `git commit -m` ?, but instead using `npm run commit` ?.

After running `npm run commit`, commitizen CLI will initialize and it will help you through. All of your commit messages will be linted by `commitlint` to make sure they're conventional.

Please note that your commit message has a direct impact on the deployment of a new release. When your PR is merged to `master`, any changes in your PR with;

\* commit(s) of type `fix` will trigger a new `patch` release, e.g. +`0.0.1`

\* commit(s) of type `feat` will trigger a new `minor` release, e.g. +`0.1.0`

\* commit(s) with `BREAKING CHANGE` in body or footer of the message will trigger a new `major` release, e.g. +`1.0.0`

You can read more about it on [commitizen](https://github.com/commitizen/cz-cli), [semver](https://semver.org), and [semantic-release](https://github.com/semantic-release/semantic-release).

All other commit types will trigger no new release. See the history of [release workflow on GitHub actions](https://github.com/onderceylan/pwa-asset-generator/actions?query=workflow%3ARelease).

## Working with TypeScript

Project is written with TypeScript language, meaning that an additional compilation step is required with `npm run build` script.

### Development

You can run TypeScript compiler - tsc in watch mode by executing `npm start`. Once you run `npm start` for the first time, tsc will create compiled JS files in /dist folder and it will watch for all TS files in /src folder, updating corresponding JS files when there's a change.

To functionally test cli over compiled cli.js code in /dist folder, you can execute the binary located in `bin` folder via your command line. Simply execute `./bin/cli [args]` in your command line to test cli functionality in development.

### Build

Run `npm run build` to compile TypeScript source to JavaScript. JS files will be created in `/dist` folder and this is the folder being published on npm during automated deployment.

## Testing

### Manual tests

If you'd like to manually test the library, you can follow the steps below;

#### Testing the functionality

1. Build the library with `npm run build` command

2. Execute the cli binary over `./bin/cli` command

#### Testing the package contents

1. Build the library with `npm run build` command

2. Pack a package for npm `npm pack` command - this will create a package in your local repo with the name `pwa-asset-generator-x.y.z.tgz`

3. This will show you the output of package creation, with the file contents of the package

#### Testing the lib e2e

1. Either install the package globally by running `npm pack && npm i -g pwa-asset-generator-x.y.z.tgz` or `npm i -g .`

2. Test the functionality with `pwa-asset-generator` command

### Automated tests

The project uses [jest](https://jestjs.io) for automated integration and e2e testing with snapshot and visual regression testing approaches. e2e tests are encouraged over unit tests in this project.

Tests can be executed with `npm test` command.

Both snapshots and visual samples can be updated with `npm run test:update` when necessary.

#### Snapshot testing

\* Utilizing jest's handy [snapshot testing](https://jestjs.io/docs/en/snapshot-testing) feature while testing feature integration is recommended. Please see [main.test.ts](https://github.com/onderceylan/pwa-asset-generator/blob/master/src/main.test.ts) file and [snapshots](https://github.com/onderceylan/pwa-asset-generator/tree/master/src/\_\_snapshots\_\_) folder for examples.

\* You can update the snapshots with `npm run test:update:snapshots` command when necessary.

#### Visual regression testing

\* There are tests in [main.test.ts](https://github.com/onderceylan/pwa-asset-generator/blob/master/src/main.test.ts) file, running visual regressions over generated image file set. It tests if generated images are visually different from samples saved in [snapshots/visual](https://github.com/onderceylan/pwa-asset-generator/tree/master/src/\_\_snapshots\_\_/visual) folder, guaranteeing the expected output for all image generation scenarios.

\* You can update the visual samples with `npm run test:update:visuals` command when necessary.

> Please apply any of your code changes alongside with e2e or integration tests.

## Continuous integration

When you create a PR for this project, it will automatically trigger a new workflow run executing automated tests on multiple platforms. You can track the build status within your PR on GitHub. You should make sure that all checks pass.