# FAQ

Please check out [the

FAQ](https://github.com/TypeStrong/atom-typescript/blob/master/docs/faq.md)

before creating a new issue :rose:

# TIP

Before doing any meaningful work or even investigating [please create an

issue for

discussion](https://github.com/TypeStrong/atom-typescript/issues) so we

don't have duplicate work and we don't step on your toes.

# Hacking on atom-typescript

This project is developed in TypeScript. TypeScript isn't directly

supported by Atom, so it requires transpilation into JavaScript. Atom

packages are just git tags, so transpiled sources have to be included

into version control.

Consequently, \*\*please avoid editing files in `dist/` directly\*\*, since

those are generated and your changes will be gone after the next build.

Also see the [Atom contributing

guide](https://github.com/atom/atom/blob/master/CONTRIBUTING.md)

## Getting started

Is rather simple. Here are some steps to get you running:

1. Fork the repository (the button in top right corner of GitHub page)

2. Clone your fork

- With SSH:

`git clone git@github.com:yourusername/atom-typescript.git`

Or

- With HTTPS:

`git clone https://github.com/yourusername/atom-typescript.git`

Either of these commands will create a working copy of the

repository in `atom-typescript` directory.

All further commands in this list are assumed to be run from root of

the working copy (i.e. `atom-typescript` directory, the one

containing `package.json`)

3. Create a new branch! `git checkout -b my-awesome-contribution`.

Please use a meaningful name for your branch.

4. Install dependencies with `apm install`.

Install development dependencies with `npm install --only=dev`.

Optionally `apm link` if you want to test your changes in Atom. Be

careful if you're using `atom-typescript` to hack on

`atom-typescript` though! You'd be hacking on the software using the

same software you're currently hacking on, which sounds somewhat

convoluted because it is. See [section on workflow](#workflow) below

for some tips.

5. Hack on it!

6. Prettify the code by running `npm run prettier`

7. Transpile to JavaScript by running `npm run build`

8. Run static checks with `npm run test` (this will run typechecker and

linter, and check if formatting is OK)

9. Run dynamic test-suite with `apm test` (at the moment, it's rather

anemic and only checks if package loads at all)

10. Commit your changes. Don't forget to commit transpiled source in

`dist/`. Write a meaningful description for your commit! Push often!

Repeat steps 4-9 until satisfied.

11. Create a pull request.

\*\*Note\*\*: feel free to create pull requests at any stage of the process.

Earlier is usually better. For one, creating PRs early is a good way of

letting people know you're working on something, which helps avoid

effort duplication. Also it will allow maintainers to chime in early and

help you avoid pitfalls and common mistakes.

## Pull

Whenever you pull in latest changes, you should run `npm install`.

Whenever we update to latest TypeScript we need to recompile all our js

to make sure everybody gets the same code.

## Git

You need to have git. Note on windows long file paths can be an issue so

run:

git config --system core.longpaths true

And use `Shift+Delete` to delete files if simple `delete` doesn't work.

# Various

## Publishing

- If you have only fixed bugs in a backward-compatible way (or

consider your changes very minimal), run `apm publish patch`.

- If you have implemented new functionality, run `apm publish minor`.

(A TypeScript update should at least be minor).

- For breaking changes run `apm publish major`. These must be

justified with a reason documented in `changelog.md`

Additional Notes:

- The `apm` command does a lot for you \*that you shouldn't do

manually\*. It automatically updates the `package.json` +

`creates a git tag` + `pushes to git` + `pushes to apm`.

- On windows : storing your github password using

`git config --global credential.helper wincred` helps smooth out the

`apm publish <type>` experience.

## Workflow

\*\*We develop atom-typescript with atom-typescript\*\*

Some shortcuts:

- `ctrl+alt+i` or `ctrl+shift+i` (View ? Developer ? Toggle Developer

Tools... menu item) will open the dev tools. These are the same

Chrome dev tools you may be familiar with. Feel free to inspect

elements. This will come handy when doing UI or even seeing why a

particular code element is highlighted in some way.

- `ctrl+alt+r` or `ctrl+shift+f5` (`window:reload` command) will

reload the entire atom instance.

### General Steps

1. We open `atom-typescript` source in one Atom window

2. We have

[`atom-typescript-examples`](https://github.com/TypeStrong/atom-typescript-examples)

open in another atom window as such: `atom --dev <examplesFolder>`

3. We make changes to `atom-typescript` and save to get the JS

(optionally run `typescript:build` command to rebuild everything)

4. We typecheck whole project with `typescript:check-all-files` command

to see if our changes accidentally broke anything.

5. We reload the `atom-typescript-examples` (`ctrl+alt+r` or

`ctrl+shift+f5`) window to see the effects of our change.

6. Only reload the `atom-typescript` window once we are sure that our

new code is functional.

### When you break atom-typescript during development

This shouldn't happen as long as you start the `atom-typescript` window

\*without\* the `--dev` flag, and do testing in another atom instance. If

you reload the `atom-typescript` window thinking its going to be stable

but it turns out to be unstable, discard \*JavaScript\* changes that you

think broke it and reload the atom instance.

For example, this will revert to last commit:

git checkout dist

And if you need to go back to `master` branch:

git checkout origin/master -- dist

## Language Service Documentation

The TypeScript Language service docs:

<https://github.com/Microsoft/TypeScript/wiki/Using-the-Compiler-API>

The `tsserver` protocol definitions

<https://github.com/Microsoft/TypeScript/blob/master/lib/protocol.d.ts>

## Showing errors in atom

Done using [Linter V2 Indie

API](https://steelbrain.me/linter/types/indie-linter-v2.html).