# Contributing to RMWC

RMWC is open source and if you're reading this, it's your job to make it better.

## Development Process

We strive to make developing RMWC as frictionless as possible, both for ourselves and for our community. This section should get you up and running working on the RMWC codebase. Before beginning development, you may want to dive into the source code to get an overall understanding of structure.

### Setting up your development environment

You'll need a recent version of [nodejs](https://nodejs.org/en/) to work on RMWC. We test using the latest node versions, so use the latest stable version where possible. The dev environment is verified to work with Node versions 7 and up. Currently, RMWC development has only been done on Mac. If you are using a windows or linux based machine, please document your experience!

> \*\*NOTE\*\*: If you expect to commit updated or new dependencies, please ensure you are using npm 5, which will

> also update `package-lock.json` correctly when you install or upgrade packages.

Once node is installed, simply clone our repo (or your fork of it) and run `npm install`.

```

git@github.com:jamesmfriedman/rmwc.git # or a path to your fork

cd rmwc && npm i

```

### Contributing to Existing Components

As of V5, RMWC has been rebuilt in Typescript which makes the developer experience much easier to navigate. `material-components-web` converted to Typescript as well which allows for a very robust and tight integration.

- Storybook is used for developing components in isolation, but the docs can be used for development as well

- The README.md in the docs are automatically generated at release time from the Typescript readme.tsx files. DO NOT UPDATE a generated-\*.json, or a README.md file fro a component manually

```jsx

class DocsExample {

state = {}

render() {

<Button>Hello</Button>

<Button

raised={this.state.raised || true}

onClick={() => this.setState({raised: !this.state.raised})}

label="Hello"

/>

}

}

```

- It is also worth noting that the foundation / adapter integration that `material-components-web` suggests using can appear daunting at times. Don't let it scare you out of trying to give back. When in doubt, just ping us on [Discord](https://discordapp.com/invite/4BSUxCW)

## Cheat sheet

- `npm install` - does all of the installing and setup

- `npm start` - run the docs site

- `npm run storybook` - develop components in storybook

- `npm run test:watch` - run the tests in watch mode while developing

- `npm run docgen` - generates the prop types for the docs. This is a very heavy process that gets run automatically on release. Only run this manually if you're trying to check to see if you updated the docs correctly.

## Typings

- As of V5, RMWC has been rewritten in Typescript.

- V4 and below supported Flow. Some remnants may remain in the codebase, but they are no longer used.

- We still want to support Flow! If you know a way forward, please comment on this issue https://github.com/jamesmfriedman/rmwc/issues/407

### Building New Components

Each component requires the following items in order to be complete:

\* A \*\*index.tsx\*\* file containing the bulk of the component code. This makes importing easier.

\* A \*\*story.tsx\*\* file using Storybook. This is where the component can be developed in isolation.

\* A \*\*test.spec.tsx\*\* file using enzyme

\* A \*\*test-ssr.spec.tsx\*\* file using enzyme testing server side rendering

\* A \*\*readme.tsx\*\* file.

\* A \*\*package.json\*\* file with the appropriate dependencies listed

\* It needs to be added to the src/rmwc/package.json

\* It needs to be re-exported from rmwc/index for people using the RMWC global

\* Its css exports should be added to src/rmwc/styles.tsx

### Running development server

#### Local development server

```

npm run start

open http://localhost:3000

```

### Building RMWC

To "build" the library for distribution

```

npm run build

```

To build the docs

```

npm run build:docs

```

#### Running Tests

Make sure you have the devDependencies installed and then just use `npm test`. This will do a single run of the entire test suite in all supported versions of React. For development, you can use `npm run test:watch`.

```

npm test OR npm run test:watch

```

### Coding Style

Our entire coding style is enforced automatically through the use of eslint and prettier. You need to use an IDE that supports these plugins or have a way to post process the files with prettier-eslint. VSCode is free and a breeze to use.

### Submitting Pull Requests

When submitting PRs, make sure you're following our commit message conventions.

RMWC uses [generate-changelog](https://www.npmjs.com/package/generate-changelog) to build the CHANGELOG.md file. All commits should only complete a single task and be formatted like the following. Use the most relevant label from generate-changelog, i.e. fix, breaking, docs, chore, etc.

```

type(category): description [flags]

fix(Button): Fixed some terrible bug

chore(Slider): Did something tedious

```

When submitting PRs for large changes, be sure to include an adequate background in the description

so that reviewers of the PR know what the changes entail at a high-level, the motivations for making

these changes, and what they affect.

If you've done some experimental work on your branch/fork and committed these via `git commit --no-verify`, you can rebase them into one correctly-formatted commit before submitting.

Finally, it helps to make sure that your branch/fork is up to date with what's currently on master. You can ensure this by running `git pull --rebase origin master` on your branch.

> \*\*NOTE\*\*: Please do \_not merge\_ master into your branch. \_Always\_ `pull --rebase` instead. This ensures a linear history by always putting the work you've done after the work that's already on master, regardless of the date in which those commits were made.

### Archiving older versions of the docs

This is mainly a cheat sheet for the maintainers. Its a manual process, but only happens on major releases.

- Pull a fresh copy or rmwc/master

- open the root package.json file

- change homepage to the version you are archiving "https://rmwc.io" to "homepage": "https://rmwc.io/version/x.x.x",

- run `npm run build:docs`

- copy the contents the docs folder to public/version/x.x.x. Make sure you remove the "sub" version folder if there is one

- edit docs/common/doc-versions to add the version x.x.x

- Do NOT commit the updates to package.json, discard it

- Commit the updated docs