# Contributing

First off, thanks for taking the time to contribute!

The following is a set of guidelines for contributing to `react-md` and its

packages. These are mostly guidelines, not rules. Use your best judgment, and

feel free to propose changes to this document in a pull request.

## Table of Contents

<!-- toc -->

- [I don't want to read the whole thing I just want to ask a question!](#i-dont-want-to-read-the-whole-thing-i-just-want-to-ask-a-question)

- [Did you find a bug?](#did-you-find-a-bug)

- [Do you have a new feature request?](#do-you-have-a-new-feature-request)

- [Do you want to fix a bug or add a new feature?](#do-you-want-to-fix-a-bug-or-add-a-new-feature)

- [Quick Start](#quick-start)

- [Initial Setup](#initial-setup)

- [Documentation Server](#documentation-server)

- [File Structure](#file-structure)

- [nextjs pages](#nextjs-pages)

- [Creating a new Demo](#creating-a-new-demo)

- [Running Tests](#running-tests)

- [What types of tests should be written?](#what-types-of-tests-should-be-written)

- [Linting](#linting)

- [Formatting](#formatting)

- [Scripts](#scripts)

<!-- tocstop -->

## I don't want to read the whole thing I just want to ask a question!<!-- force-heading -->

If you have a question about how to use `react-md`, getting started, or general

styling questions, please do not create an issue. Instead, join the [official

Slack channel] where someone will hopefully answer your question for you.

> You will need an invite to join the official Slack channel, but luckily you

> can get an invite by entering your email into the automated invite site:

> https://react-md.herokuapp.com

>

> Note: this invite site is running on a heroku free tier plan, so if you try to

> get an invite near the end of the month, this site might be down. If you need

> to join immediately, you can send an email to mlaursen03@gmail.com with a

> subject of "react-md Slack invite". You can also use this [email template].

## Did you find a bug?

If you found a bug, \*\*first\*\* search the existing issues to see if it has

already been reported. If the bug has not been reported, create an issue on

GitHub using the bug report template and provide as much information as

possible. Attaching screenshots as well as a runnable code example using

[codesandbox] will help get a bugfix out quicker due to less debugging and

initial setup.

When creating the issue name, please be concise and focus on a specific package,

version, or browser that this occurs in. Here is a list of good bug issue

titles:

- The menu component does not render correctly in Firefox

- Applying a dynamic theme in mobile Safari does not work after version x

- The `ComponentName` in `@react-md/package` is not accessible

## Do you have a new feature request?

If you would like to see a new feature within `react-md`, please open an issue

on GitHub using the feature request template. To help the contributors

prioritize a feature, please provide as much information as possible any helpful

screenshots, or a link to a page on the [material design] website.

Since this project is really maintained by only one developer, the scope of

allowed features will be limited to:

- something related to material design

- adding additional accessibility

- ease-of-use to existing components

## Do you want to fix a bug or add a new feature?

Before you start working on a bug or feature, please make sure that there has

already been an issue filed and the fix/feature has been approved as a valid bug

or feature. From there, you'll want to follow the steps below which will outline

initial setup and other information.

## Quick Start

If you just want to get quickly started and run the documentation server, you

can run the following commands:

```sh

$ yarn

$ yarn setup

$ yarn dev

```

This will install all the packages, run the build for the first time, and start

up the development server in watch mode.

If this is your first time contributing, I really recommend reading the next few

points for more information before getting started.

## Initial Setup

To get started, you'll first want to fork the [react-md repo] and then clone

your forked repository locally. Once the project has been cloned, you'll want to

install the dependencies and get an initial build rolling. `react-md` is

developed as a monorepo using [yarn] and [lerna] and might not work with `npm`,

so you will want to install `yarn` first. Once yarn is installed, you can also

optionally install `lerna` globally or use [npx] for all `lerna` commands.

Installing `lerna` is optional since the [base npm scripts] will cover most of

the cases for you and use the locally installed `lerna` by default.

> `npx` is available with `npm@5.2` or greater

Now that `yarn` (and optionally `lerna`) have been installed, you can install

and bootstrap all the packages using `yarn` or `yarn install`.

Next, you'll need to build the `dev-utils` package to be able to build each

project and have some helpful utils available while developing `react-md`:

```sh

$ yarn dev-utils

$ yarn --force

```

> Note the `yarn --force`. This is unfortunately required since the `dev-utils`

> package creates an executable and for some reason `yarn` can't symlink it

> correctly if you ran install before this executable existed. Running

> `yarn --force` after the `dev-utils` package is built allows for it to be

> available in all the other packages.

Now that the `dev-utils` have been installed and are ready to use, it's time to

run your first build. Unfortunately, this will take a bit of time the first time

it is run (~5-10 min). Start the build with:

```sh

$ yarn build

```

Whew! We're getting there. Now that you have your initial build, you can start

developing and start up the documentation server! You can start the server with:

```sh

$ yarn start

```

Or

```sh

$ yarn dev

```

Using `yarn start` will only start up the documentation server while `yarn dev`

will also start up the `yarn watch` command which will automatically update any

scss variables that have been changed and create a Typescript watcher for the

`react-md` packages.

## Documentation Server

The documentation server is built using [nextjs] to provide server side

rendering and runs using a custom typescript server. The app will be available

at http://localhost:3000 by default, but you can configure the port by setting

the `PORT` environment variable:

```sh

$ PORT=8080 yarn start

```

### File Structure

The documentation folder will contain the following folders:

- `public` - Static files to be served. This shouldn't really include many other

files as most of the time the static files should be imported normally with

webpack.

- `src/components` - All the React components to be used throughout the app.

This is 95% of the codebase

- `src/constants` - Any general constants to use throughout the app and demos.

This is where reusable data should be stored

- `src/guides` - A list of all the `\*.md` files that should be used as guides

- `src/hooks` - Any reusable hooks for the documentation site only

- `src/icons` - Any custom icons that are not available from Material Icons

- `src/pages` - All the pages in the site. Check out the next block for more

information

- `src/server` - The custom typescript server

- `src/types` - Any custom types needed to run the app

- `src/utils`- General utils used for the documentation site. This include

things like a simple number generator or some string utils

### nextjs pages

If you haven't used `nextjs` before, the way it works is that every file found

in the `src/pages` directory will be turned into a route in the website. So if

we want to have the `/customization` and `/customization/color-palette` routes,

you'll have a file structure of:

```

src/pages/customization/index.tsx

src/pages/customization/color-palette.tsx

```

Since this can lead to some pages accidentally being created, components aren't

actually created in this directory and instead are found within the base

`src/components` folder. All the files within the `src/pages` folder will just

be a simple \_pointer\_ to a file in the `src/components` folder. A really good

example for this is how the About page is created. All that exists at

`src/pages/about.ts` is a single line:

```ts

export { default } from "components/About";

```

### Creating a new Demo

This will be where all the package demos are stored and where 90% of the

contributions and development will be. You'll find a all the packages within

`react-md` in this directory with `PascalCase`. Each of these folders will

contain:

- an `index.tsx`

- `DemoName.tsx`

- `DemoName.md`

The main `index.tsx` file will include all the demos within the app and just

render a `DemoPage` component. Each demo will at least need:

- a name

- a description

- `children`

Where the description should be the `DemoName.md` file, the children should be

the `DemoName.tsx`, and the name should be the `Demo Name`.

It'll probably be the most helpful to just check out the [Demo source code] as

well as an [example demo file] to get a more in-depth example.

If your demo requires more than one additional file, you should create a

directory for all the demo files including the main readme.

```

??? CustomRenderers

?   ??? CustomRenderers.tsx

?   ??? NonVirtualizedMenu.tsx

?   ??? README.md

?   ??? VirtualizedMenu.tsx

?   ??? index.ts

```

## Running Tests

Tests are being run with [jest] and using the [react-testing-library] to test

React components.

You can run tests at the root level to test all packages:

```sh

mlaursen @ ~/code/react-md

$ yarn test

```

The test command also supports the `--watch` flag so you can continually run

tests while developing.

### What types of tests should be written?

TODO:

## Linting

The Typescript and JavaScript files found within this repo will be linted with

[eslint] while the SCSS files are handled by [sass-lint]. If your IDE or text

editor supports these linters, errors should be caught in real-time while

editing files. Otherwise, you can manually trigger the lint command with:

```sh

mlaursen @ ~/code/react-md

$ yarn lint

# or for typescript only

$ yarn lint-ts

# or for scss only

$ yarn lint-styles

```

## Formatting

This project is currently formatted using [prettier], so it is recommended to

add support in your IDE to automatically format as needed. Otherwise, all files

will automatically be formatted pre-commit.

## Scripts

The most important scripts while developing and contributing to react-md will be

the `start`, `test`, and `watch` commands. Running `yarn start` will start up

the documentation server using [nextjs] and the `watch` command will dynamically

start up `tsc` watchers for each package when a file change is detected as well

as re-distributing any changed scss files. If you want to see any additional

useful dev utils and scripts, run:

```bash

$ yarn dev-utils --help

```

[lerna]: https://lernajs.io/

[yarn]: https://yarnpkg.com

[typescript]: https://www.typescriptlang.org/

[codesandbox]: https://codesandbox.io

[official slack channel]: https://react-md.slack.com

[email template]: mailto:mlaursen03@gmail.com?subject=react-md%20Slack%20invite

[material design]: https://material.io/develop/web

[react-md repo]: https://github.com/mlaursen/react-md

[nextjs]: https://github.com/zeit/next.js

[npx]:

https://medium.com/@maybekatz/introducing-npx-an-npm-package-runner-55f7d4bd282b

[base npm scripts]: #scripts

[jest]: https://github.com/facebook/jest

[react-testing-library]:

https://github.com/testing-library/react-testing-library

[enzyme]: https://airbnb.io/enzyme/

[eslint]: https://github.com/eslint/eslint

[prettier]: https://github.com/prettier/prettier

[sass-lint]: https://github.com/sasstools/sass-lint

[demo source code]:

https://github.com/mlaursen/react-md/tree/master/packages/documentation/src/components/Demos/Demo.tsx

[example demo file]:

https://github.com/mlaursen/react-md/tree/master/packages/documentation/src/components/Demos/AppBar/index.tsx