# Contributing to Razzle

Hi there! Thanks for your interest in Razzle. This guide will help you get started contributing.

<!-- START doctoc generated TOC please keep comment here to allow auto update -->

<!-- DON'T EDIT THIS SECTION, INSTEAD RE-RUN doctoc TO UPDATE -->

\*\*Table of Contents\*\*

- [Contributing to Razzle](#contributing-to-razzle)

- [Project Structure](#project-structure)

- [Developing locally](#developing-locally)

- [Commands](#commands)

- [Updating your fork](#updating-your-fork)

- [Adding examples](#adding-examples)

- [Use `examples/basic` as template](#use-examplesbasic-as-template)

- [Naming examples](#naming-examples)

- [How to get your example merged](#how-to-get-your-example-merged)

- [Why wasn't my PR merged?](#why-wasnt-my-pr-merged)

- [Getting help](#getting-help)

<!-- END doctoc generated TOC please keep comment here to allow auto update -->

## Project Structure

Razzle is monorepo made up of a several npm packages powered by Lerna.

- `examples`: All examples go in here.

- `packages`: This is where the magic happens

- `babel-preset-razzle`: Razzle's default Babel preset.

- `create-razzle-app`: Razzle's CLI tool responsible for initialization of new projects

- `razzle`: The core library

- `razzle-dev-utils`: Utilities and helpers

- `scripts`: Utility scripts related to cleaning and bootstrapping the repo

- `test`: End-to-end tests

## Developing locally

First, fork the repo to your GitHub account. Then clone your fork to your local

machine and make a new branch for your feature/bug/patch etc. It's a good idea to not develop directly on master so you can get updates.

```

git clone https://github.com/<YOUR\_GITHUB\_USERNAME>/razzle.git

cd razzle

git checkout -B <my-branch>

yarn

```

This will install all `node\_modules` in all the packages and all the examples and symlink

inter-dependencies. Thus when you make local changes in any of the packages you can try them

immediately in all the examples.

### Commands

- `yarn run clean`: Clean up all `node\_modules` and remove all symlinks from packages and examples.

- `yarn run bootstrap`: Run `yarn` on all examples and packages. Automatically symlinks inter-dependent modules.

- `yarn run test`: Runs all tests

- `yarn run test:packages`: Runs tests for packages

- `yarn run test:e2e`: Runs end-to-end tests

### Updating your fork

When you want to pull down changes to your fork enter the following into your terminal:

```bash

git checkout master

git pull origin master

```

## Adding examples

### Use `examples/basic` as template

If you'd like to add an example, I suggest you duplicate the `examples/basic` folder and use that as kind of base template. Before you start adding stuff, go ahead and change the name of the package in the your new example's `package.json`. Then go back to the project root and run `yarn bootstrap`. This will make sure that your new example is using your local version of all the `packages`.

### Naming examples

All example folders should be named `with-<thing-you-are-demonstrating>`. Each example's npm package name (found in it's `package.json`) should look like `razzle-examples-with-<thing-you-are-demonstrating>`.

### How to get your example merged

- Make sure to comment the important parts of your code and include a \*\*well-written\*\*

"Idea behind the example" section. This is more important to me than your actual code.

- Keep your example limited to one idea / library / feature (e.g. don't submit `with-styled-components-and-material-ui`). That being said, there are times when this rule will be relaxed such as if you are showing how to use Apollo and Redux or \<Flux Library\> + React Router.

- Your example \*\*MUST\*\* implement Hot Module Replacement. If it does not update when you make edits, you have broken something.

- Your example should be minimalistic and concise, or a direct copy of another prominent example from the original library (like copying an example directly from react-redux).

## Why wasn't my PR merged?

I will do my best to write out my reasoning before closing a PR, but 80% of the time it falls under one of these...

- You did not read this document

- Your code breaks an internal application (I will be transparent about this)

- Your code conflicts with some future plans (I will be transparent about this too)

- You've said something inappropriate or have broken the Code of Conduct

## Getting help

Tweet / DM [@jaredpalmer](https://twitter.com/jaredpalmer)