# Contributing Guide

Contributions are welcome and are greatly appreciated! Every little bit helps, and credit will

always be given.

## Setting up your environment

After forking enzyme to your own github org, do the following steps to get started:

```bash

# clone your fork to your local machine

git clone https://github.com/enzymejs/enzyme.git

# step into local repo

cd enzyme

# install dependencies (use `react 13` if you want to use React 0.13)

npm install

# install react version

# accepts `13` for v0.13, `14` for v0.14, and for versions 15+,

# accepts either a major (`15`, `16`) or a minor (`15.4`, `16.8`)

npm run react 16

```

### Switching between React 16, React 15, React 0.14 and React 0.13

```bash

# switch to React 0.13

npm run react 13

```

```bash

# switch to React 0.14

npm run react 14

```

```bash

# switch to React 15

npm run react 15

```

```bash

# switch to React 16

npm run react 16

```

Specific versions can also be specified

```bash

# switch to React 16.5

npm run react 16.5

```

### Running Tests

The test suite runs on \*built\* Enzyme.

```bash

# build Enzyme locally before testing

npm run build

# run tests on whatever version of React is currently installed

npm test

```

```bash

# run tests on all supported versions of React

npm run test:all

```

If you are actively developing, Enzyme will always need to be built with the latest changes.

For this, the recommended workflow is to have the build and tests watching for changes in separate terminals. This should provide you with ~realtime feedback:

```bash

# build Enzyme locally upon save

npm run build:watch

# faster feedback for TDD

npm run test:watch

```

### Tests for functionality shared between `shallow` and `mount`

Tests for a method "foo" are stored in `packages/enzyme-test-suite/test/shared/methods/foo`. The file default exports a function that receives an injected object argument, containing the following properties:

- `Wrap`: e.g. `shallow`, `mount`

- `WrapRendered`: this abstracts around the differences between `shallow` and `mount` - e.g., that the root of a shallow wrapper around `Foo` is what `Foo` \*renders\*, where the root of a mount wrapper around `Foo` is `Foo` itself. Thus, this function produces a wrapper around what `Foo` renders, regardless of the `Wrap` method used.

- `Wrapper`: e.g. `ShallowWrapper`, `ReactWrapper`

- `WrapperName`: e.g. `"ShallowWrapper"`, `"ReactWrapper"`

- `isShallow`: true if `shallow`. note: needing to use this is a code smell, please avoid.

- `isMount`: true if `mount`. note: needing to use this is a code smell, please avoid.

- `makeDOMElement`: in `mount`, makes a real DOM element; in `shallow`, makes a mock object.

These tests are ran via an explicit list in a `describeMethods` call in the ReactWrapper and ShallowWrapper test files. If you add a new test file for a shared method, you'll need to add its name to both calls.

### Style & Linting

This codebase adheres to the [Airbnb Styleguide](https://github.com/airbnb/javascript) and is

enforced using [ESLint](http://eslint.org/).

As with the test suite, the linter will not fully pass unless it is running on \*built\* Enzyme. This is because the ESLint `import/\*` rules rely on finding the target files in the filesystem (which won't be there unless they've been built).

It is recommended that you install an ESLint plugin for your editor of choice when working on this

codebase, however you can always check to see if the source code is compliant by running:

```bash

# build Enzyme locally before linting

npm run build

npm run lint

```

### Publishing

Enzyme uses [lerna](https://github.com/lerna/lerna) to structure its repo, and has multiple packages

to publish out of this one repo. We use lerna's "independent" mode, which means that the versioning

of each package in the repo is versioned independently.

We are waiting on [this issue](https://github.com/lerna/lerna/issues/955) to be fixed, so that

`peerDependencies` do not get updated with patch updates.

Until this issue is fixed, we will publish each package manually instead of with `lerna publish`. In

order to do this, we will:

For enzyme:

```bash

# ... update version in enzyme/package.json, make changes to CHANGELOG, etc.

cd packages/enzyme

git commit -m v{version}

git tag -a -m v{version}

git push --follow-tags

npm publish

```

For other packages

```bash

# ... update version in {package}/package.json, make changes to CHANGELOG, etc.

cd packages/{package}

git commit -m "{package}: v{version}"

git tag -a -m "{package}: v{version}"

git push --follow-tags

npm publish

```

Once we are able to use `lerna publish`, the process will be as follows:

Lerna by default will only publish packages that have changed since the last release. It will also

create a tagged commit for each release.

To publish, run:

```bash

lerna publish -m "{tag name}"

```

The tag name is determined by the `-m` CLI option. If `enzyme` is one of the packages that has

updates, we default to just using that version as the tag name. For instance, when publishing

`enzyme@3.1.1` and `enzyme-adapter-react-16@1.2.3` we would run:

```bash

lerna publish -m "v3.1.1"

```

If `enzyme` is \*not\* one of the packages being updated, use the other package's name and the version:

```bash

lerna publish -m "enzyme-adapter-react-16: v1.2.3"

```

The `lerna publish` command will present interactive prompts asking which version to use for each

package independently. Just choose whichever

### Building Docs

Building the docs locally is extremely simple. First execute the following command:

```bash

npm run docs:watch

```

After this, you can open up your browser to the specified port (usually http://localhost:4000 )

The browser will automatically refresh when there are changes to any of the source files.

## Pull Request Guidelines

Before you submit a pull request from your forked repo, check that it meets these guidelines:

1. If the pull request fixes a bug, it should include tests that fail without the changes, and pass

with them.

1. If the pull request adds functionality, the docs should be updated as part of the same PR.

1. The pull request should work for React 15, React 0.14 and React 0.13. The CI server should run the

tests in all versions automatically when you push the PR, but if you'd like to check locally, you

can do so (see above).

1. Please rebase and resolve all conflicts before submitting.