## Hey there!

So you want to know more about React Cosmos... \_Sweet!\_

> You're probably a nice person and don't need to hear this. At the same time, however, you \_could\_ be one of the assholes roaming the world. So please follow our [Code of Conduct](CODE\_OF\_CONDUCT.md) and respect the people you interact with.

Jump to:

- [Goals](#goals)

- [How to contribute](#how-to-contribute)

- [Repo](#repo)

## Goals

\*\*Make UI developers more productive.\*\*

This is \_the\_ goal. The means are built on the following insight: Designing components in isolation yields reusable components. Reusability is at the heart of productivity.

\*\*Help UI developers think long term.\*\*

UI development is all fun and games at first. But sooner or later it gets messy. \_Massively messy.\_ Why? Because calling them "components" isn't enough to avoid tight coupling. Developing components in isolation discourages coupling.

\*\*Make UI development fun.\*\*

As UI developers, we spend our days improving other people's digital experience. \_What about us?\_ Let's treat ourselves with the same delightful experience we've accustomed our users to. Something something virtuous circle.

## How to contribute

It's common to only think of open source contributions as code. But if you're new to a project submitting a PR should almost never be the first thing you do. Even the most organized codebases have a learning curve. Learn how a project works before trying to change it.

Here's a list of things you can do to help React Cosmos, sorted by project exprience required:

1. \_Use\_ the project. Understand its capabilities.

2. Provide user feedback.

3. Propose ideas to improve onboarding and user experience – not just features :).

4. Respond to issues you know how to solve.

5. Engage with the community.

6. Engage in roadmap & design discussions.

7. Fix known bugs.

8. Implement agreed upon changes.

### Be kind and thoughtful

We're all short on time, so be realistic and don't expect special treatment. The better we communicate the more likely we'll end up collaborating.

### Ask for review

? \*\*[Use the RFCs process for substantial changes](https://github.com/react-cosmos/rfcs)\*\*

\*\*Please propose an idea before coding it.\*\* Otherwise your work might get rejected, which is never fun. Save everybody's time by asking for feedback \_before\_ implementing something.

### CI failed on your PR?

Open the CI build page. See what went wrong and learn to run the checks locally. Don't expect a review if your build is broken. Ask for help if you can't figure it out.

## Repo

> Working on Cosmos requires Node 8 or newer

[Yarn Workspaces](https://yarnpkg.com/lang/en/docs/workspaces/) and [Lerna](https://github.com/lerna/lerna) make it possible to publish independent packages and still test the project end-to-end.

Tools are installed globally in the root node\_modules. This includes Jest, TypeScript, Webpack, and their corresponding plugins. ESLint is also applied globally. Creating a new package has less overhead because of this. React and Webpack deps are also installed once in the root node\_modules to avoid version conflicts or bundling multiple copies of React.

```bash

git clone git@github.com:react-cosmos/react-cosmos.git

cd react-cosmos

# Install deps and link packages

yarn

# Basic commands

yarn check-types

yarn test

yarn build

# Run example and test React Cosmos end to end

yarn start

# Cosmos #inception

yarn start:playground

# Build single package in watch mode

yarn build react-cosmos-playground2 --watch

# Test watch mode

yarn test:watch

# Single test watch mode

yarn test:watch path/to/my/testfile

```

### Test your work

In most cases write automated tests. TDD where applicable, but don't force it. Write concise, maintainable tests. Tests should be as readable as source code.

Look inside `\_\_tests\_\_` folders and files ending with `.test` for test examples.

### Stay consistent

Add [ESLint](https://eslint.org/docs/user-guide/integrations#editors) to your editor if possible.

When naming files:

- Use \_camel case\_ for files: `fixtureState.js`. Capitalize components: `DragHandle.js`.

- Use \_kebab case\_ for package names: `react-cosmos-playground2`.

When creating a module:

- \*\*Named exports\*\* are preferred over default exports.

- \*\*Function declarations\*\* are preferred over arrow functions at the module level (because the order doesn't matter when using the former).

---

\_Have a question or idea to share? See you on [Slack](https://react-cosmos.slack.com/join/shared\_invite/zt-g9rsalqq-clCoV7DWttVvzO5FAAmVAw).\_