Thanks for contributing, you rock!

When it comes to open source, there are many different kinds of contributions that can be made, all of which are valuable. Below are a few guidelines that should help you as you prepare your contribution.

- [Think You Found a Bug?](#bug)

- [Proposing New or Changed API?](#api)

- [Issue Not Getting Attention?](#attention)

- [Making a Pull Request?](#pr)

- [Setup](#setup)

- [Development](#dev)

<a name="bug"/></a>

## Think You Found a Bug?

Please conform to the issue template and provide a clear path to reproduction with a code example. Best is a pull request with a failing test. Next best is a link to CodeSandbox or repository that illustrates the bug.

You may wish to use [this starter CodeSandbox](https://codesandbox.io/s/react-router-v5-starter-4g9ei) to help you get going.

<a name="api"/></a>

## Proposing New or Changed API?

Please provide thoughtful comments and some sample code. Proposals without substance will be closed.

It's generally a good idea to open an issue for the proposal first before working on the implementation and submitting a pull request. Please also take a look at [our current roadmap](https://github.com/ReactTraining/react-router/issues/6885) and consider ongoing work that might conflict with your proposed changes.

<a name="attention"/></a>

## Issue Not Getting Attention?

If you need a bug fixed and nobody is fixing it, it is greatly appreciated if you provide a fix for it. Issues with no activity for 60 days will be automatically closed, with a warning 7 days before closing.

<a name="pr"/></a>

## Making a Pull Request?

Pull requests need only the :+1: of two or more collaborators to be merged; when the PR author is a collaborator, that counts as one.

### Tests

All commits that fix bugs or add features need a test.

`<blink>`Do not merge code without tests.`</blink>`

### Docs + Examples

All commits that change or add to the API must be done in a pull request that also updates all relevant examples and docs.

<a name="setup"/></a>

## Setup

The following steps will get you setup to contribute changes to this repo:

1. Fork the repo (click the <kbd>Fork</kbd> button at the top right of [this page](https://github.com/ReactTraining/react-router))

2. Clone your fork locally

```bash

# in a terminal, cd to parent directory where you want your clone to be, then

git clone https://github.com/<your\_github\_username>/react-router.git

cd react-router

```

3. Install dependencies and build. React Router uses `yarn`, so you should too. If you install using `npm`, unnecessary `package-lock.json` files will be generated.

```bash

yarn install

yarn build

```

<a name="dev"/></a>

## Development

### Packages

React Router uses a monorepo to host code for multiple packages. These packages live in the `packages` directory.

React Router uses Lerna to manage the monorepo. Lerna sets up symlinks between the packages, so when you build one package, its changes will be automatically available to the packages that depend on it.

### Building

Calling `yarn build` from the root directory will build all packages. If you want to build a specific package, you should `cd` into that directory.

```bash

# build all packages

yarn build

# build react-router-dom

cd packages/react-router-dom

yarn build

```

### Testing

Calling `yarn test` from the root directory will run \*\*every\*\* package's tests. If you want to run tests for a specific package, you should `cd` into that directory.

```bash

# run all tests

yarn test

# react-router-dom tests

cd packages/react-router-dom

yarn test

```

React Router uses Jest to run its tests, so you can provide the `--watch` flag to automatically re-run tests when files change.

### Website

The code for the documentation website lives in the `website` directory. To start a development server on `http://localhost:8080` that will watch for changes, do:

```bash

cd website

yarn install

yarn start

```