# Contributing to Storybook

Thanks for your interest in improving Storybook! We are a community-driven project and welcome contributions of all kinds: from discussion to documentation to bugfixes to feature improvements.

Please review this document to help to streamline the process and save everyone's precious time.

This repo uses yarn workspaces, so you should install `yarn` as the package manager. See [installation guide](https://yarnpkg.com/en/docs/install).

## Issues

No software is bug-free. So, if you got an issue, follow these steps:

- Search the [issue list](https://github.com/storybookjs/storybook/issues?utf8=%E2%9C%93&q=) for current and old issues.

- If you find an existing issue, please UPVOTE the issue by adding a "thumbs-up reaction". We use this to help prioritize issues!

- If none of that is helping, create an issue with the following information:

- Clear title (shorter is better).

- Describe the issue in clear language.

- Share error logs, screenshots and etc.

- To speed up the issue fixing process, send us a sample repo with the issue you faced:

### Testing against `master`

To test your project against the current latest version of storybook, you can clone the repository and link it with `yarn`. Try following these steps:

#### 1. Download the latest version of this project, and build it:

```sh

git clone https://github.com/storybookjs/storybook.git

cd storybook

yarn bootstrap

```

> NOTE: on windows you may need to run `yarn` before `yarn bootstrap`!

The bootstrap command might ask which sections of the codebase you want to bootstrap. Unless you're going to work with ReactNative or the Documentation, you can keep the default.

You can also pick directly from CLI:

```sh

yarn bootstrap --core

```

#### 2a. Run unit tests

You can use one of the example projects in `examples/` to develop on.

This command will list all the suites and options for running tests.

```sh

yarn test

```

The options for running tests can be selected from the cli or be passed to `yarn test` with specific parameters. Available modes include `--watch`, `--coverage`, and `--runInBand`, which will respectively run tests in watch mode, output code coverage, and run selected test suites serially in the current process.

You can use the `--update` flag to update snapshots or screenshots as needed.

You can also pick suites from CLI. Suites available are listed below.

##### Core & Examples Tests

`yarn test --core`

This option executes tests from `<rootdir>/app/react`, `<rootdir>/app/vue`, and `<rootdir>/lib`.

Before the tests are run, the project must be bootstrapped with core. You can accomplish this with `yarn bootstrap --core`

##### CRA-kitchen-sink - Image snapshots using Storyshots

`yarn test --image`

This option executes tests from `<rootdir>/examples/official-storybook`

In order for the image snapshots to be correctly generated, you must have a static build of the storybook up-to-date :

```sh

cd examples/official-storybook

yarn build-storybook

cd ../..

yarn test --image

```

Puppeteer is used to launch and grab screenshots of example pages, while jest is used to assert matching images. (just like integration tests)

#### 2b. Run e2e tests for CLI

If you made any changes to the `lib/cli` package, the easiest way to verify that it doesn't break anything is to run e2e tests:

```sh

yarn test --cli

```

This will run a bash script located at `lib/cli/test/run\_tests.sh`. It will copy the contents of `fixtures` into a temporary `run` directory, run `getstorybook` in each of the subdirectories, and check that storybook starts successfully using `yarn storybook --smoke-test`.

After that, the `run` directory content will be compared with `snapshots`. You can update the snapshots by passing an `--update` flag:

```sh

yarn test --cli --update

```

In that case, please check the git diff before committing to make sure it only contains the intended changes.

#### 2c. Run Linter

We use eslint as a linter for all code (including typescript code).

All you have to run is:

```sh

yarn lint

```

It can be immensely helpful to get feedback in your editor, if you're using VsCode, you should install the `eslint` plugin and configure it with these settings:

```plaintext

"eslint.autoFixOnSave": true,

"eslint.packageManager": "yarn",

"eslint.options": {

"cache": true,

"cacheLocation": ".cache/eslint",

"extensions": [".js", ".jsx", ".mjs", ".json", ".ts", ".tsx"]

},

"eslint.validate": [

"javascript",

"javascriptreact",

{"language": "typescript", "autoFix": true },

{"language": "typescriptreact", "autoFix": true }

],

"eslint.alwaysShowStatus": true

```

This should enable auto-fix for all source files, and give linting warnings and errors within your editor.

### 2d. Run Cypress tests

First make sure the repo is bootstrapped.

Then run `yarn build-storybooks`, this creates a static website from all examples.

Then run `yarn serve-storybooks`, this will run the static site on the port cypress expects.

Then run `yarn add cypress -W --optional`. When this has completed cypress should be installed on your system. If it is already on your system, this step can be skipped.

Then run `yarn cypress open` if you want to see the tests run in the UI, or `yarn cypress run` to run the tests headless.

### Reproductions

#### In the monorepo

The best way to help figure out an issue you are having is to produce a minimal reproduction against the `master` branch.

A good way to do that is using the example `cra-kitchen-sink` app embedded in this repository:

```sh

# Download and build this repository:

git clone https://github.com/storybookjs/storybook.git

cd storybook

yarn bootstrap --core

# NOTE: on windows you may need to run `yarn` before `yarn bootstrap`!

# make changes to try and reproduce the problem, such as adding components + stories

cd examples/cra-kitchen-sink

yarn storybook

# see if you can see the problem, if so, commit it:

git checkout "branch-describing-issue"

git add -A

git commit -m "reproduction for issue #123"

# fork the storybook repo to your account, then add the resulting remote

git remote add <your-username> https://github.com/<your-username>/storybook.git

git push -u <your-username> master

```

If you follow that process, you can then link to the GitHub repository in the issue. See <https://github.com/storybookjs/storybook/issues/708#issuecomment-290589886> for an example.

\*\*NOTE\*\*: If your issue involves a webpack config, create-react-app will prevent you from modifying the \_app's\_ webpack config, however, you can still modify storybook's to mirror your app's version of the storybook. Alternatively, use `yarn eject` in the CRA app to get a modifiable webpack config.

#### Outside the monorepo

Sometimes your storybook is deeply ingrained in your own setup and it's hard to create a minimal viable reproduction somewhere else.

Inside the storybook repo we have a script that allows you to test the packages inside this repo in your own separate project.

You can use `npm link` on all packages, but npm linking is cumbersome and has subtle differences from what happens in a registry-based installation.

So the way our script works is that it:

- sets up a npm registry running on your own local machine

- changes your default registry to this local one

- builds all packages in the storybook repo

- publishes all packages as latest

Our script leaves the local registry running, for \*\*as long as you keep it running\*\* you can install storybook packages from this local registry.

- Navigate to your own project and then change `package.json` so the storybook packages match the version of the one you just published.

- Then you can install using `yarn` or `npm`

- Start using your storybook as normally.

If you've made a change to storybook's codebase and would want this change to be reflected in your app:

- Ensure the storybook packages are transpiled, by either having run `yarn dev` or `yarn bootstrap --core`.

- Go to the terminal where the local regitry is running and press `<Enter>`. This will kick off a new publish.

- Run the install procedure again in your local repo, (you may need to clean out node\_modules first).

- Restart your storybook.

### Updating Tests

Before any contributions are submitted in a PR, make sure to add or update meaningful tests. A PR that has failing tests will be regarded as a Work in Progress and will not be merged until all tests pass.

When creating new unit test files, the tests should adhere to a particular folder structure and naming convention, as defined below.

```sh

# Proper naming convention and structure for js tests files

+-- parentFolder

| +-- [filename].js

| +-- [filename].test.js

```

## Pull Requests (PRs)

We welcome all contributions. There are many ways you can help us. This is few of those ways:

Before you submit a new PR, make sure you run `yarn test`. Do not submit a PR if tests are failing. If you need any help, the best way is to [join the discord server and ask in the maintenance channel](https://discord.gg/sMFvFsG).

### Reviewing PRs

\*\*As a PR submitter\*\*, you should reference the issue if there is one, include a short description of what you contributed and, if it is a code change, instructions for how to manually test out the change. This is informally enforced by our [PR template](https://github.com/storybookjs/storybook/blob/master/.github/PULL\_REQUEST\_TEMPLATE.md). If your PR is reviewed as only needing trivial changes (e.g. small typos etc), and you have commit access then you can merge the PR after making those changes.

> NOTE: Although the latest stable version of storybook corresponds to the `master` branch, nearly all Storybook development happens in the `next` branch. If you submit a PR, branch off `next` and target your PR to `next`.

\*\*As a PR reviewer\*\*, you should read through the changes and comment on any potential problems. If you see something cool, a kind word never hurts either! Additionally, you should follow the testing instructions and manually test the changes. If the instructions are missing, unclear, or overly complex, feel free to request better instructions from the submitter. Unless the PR is tagged with the `do not merge` label, if you approve the review and there is no other required discussion or changes, you should also go ahead and merge the PR.

## Issue Triage

If you are looking for a way to help the project, triaging issues is a great place to start. Here's how you can help:

### Responding to issues

Issues that are tagged `question / support` or `needs reproduction` are great places to help. If you can answer a question, it will help the asker as well as anyone who has a similar question. Also in the future if anyone has that same question they can easily find it by searching. If an issue needs reproduction, you may be able to guide the reporter toward one, or even reproduce it yourself using [this technique](https://github.com/storybookjs/storybook/blob/master/CONTRIBUTING.md#reproductions).

### Triaging issues

Once you've helped out on a few issues, if you'd like triage access you can help label issues and respond to reporters.

We use the following label scheme to categorize issues:

- \*\*type\*\* - `bug`, `feature`, `question / support`, `discussion`, `dependencies`, `maintenance`.

- \*\*area\*\* - `addon: x`, `addons-api`, `stories-api`, `ui`, etc.

- \*\*status\*\* - `needs reproduction`, `needs PR`, `in progress`, etc.

All issues should have a `type` label. `bug`/`feature`/`question`/`discussion` are self-explanatory. `dependencies` is for keeping package dependencies up to date. `maintenance` is a catch-all for any kind of cleanup or refactoring.

They should also have one or more `area`/`status` labels. We use these labels to filter issues down so we can see all of the issues for a particular area, and keep the total number of open issues under control.

For example, here is the list of [open, untyped issues](https://github.com/storybookjs/storybook/issues?utf8=%E2%9C%93&q=is%3Aissue%20is%3Aopen%20-label%3A%22bug%22%20-label%3A%22discussion%22%20-label%3A%22feature%22%20-label%3A%22maintenance%22%20-label%3A%22question%20%2F%20support%22%20-label%3A%22documentation%22%20-label%3A%22greenkeeper%22), or here is a list of [bugs that have not been modified since 2017-04-01](https://github.com/storybookjs/storybook/issues?utf8=%E2%9C%93&q=is%3Aissue%20is%3Aopen%20label%3A%22bug%22%20updated%3A%3C%3D2017-04-01%20). For more info see [searching issues](https://help.github.com/articles/searching-issues/) in the Github docs.

If an issue is a `bug`, and it doesn't have a clear reproduction that you have personally confirmed, label it `needs reproduction` and ask the author to try and create a reproduction, or have a go yourself.

### Closing issues

- Duplicate issues should be closed with a link to the original.

- Unreproducible issues should be closed if it's not possible to reproduce them (if the reporter drops offline,

it is reasonable to wait 2 weeks before closing).

- `bug`s should be labelled `merged` when merged, and be closed when the issue is fixed and released.

- `feature`s, `maintenance`s, `greenkeeper`s should be labelled `merged` when merged,

and closed when released or if the feature is deemed not appropriate.

- `question / support`s should be closed when the question has been answered.

If the questioner drops offline, a reasonable period to wait is two weeks.

- `discussion`s should be closed at a maintainer's discretion.

## Development Guide

### Prerequisites

Please have the \*\*\_latest\_\*\* stable versions of the following on your machine

- node

- yarn

### Initial Setup

If you run into trouble here, make sure your node, npm, and \*\*\_yarn\_\*\* are on the latest versions (yarn at least v1.3.2).

1. `cd ~` (optional)

2. `git clone https://github.com/storybookjs/storybook.git` \_bonus\_: use your own fork for this step

3. `cd storybook`

4. `yarn bootstrap --core`

5. `yarn test --core`

6. `yarn dev` \_You must have this running for your changes to show up\_

> NOTE: on windows you may need to run `yarn` before `yarn bootstrap` (between steps 3 and 4).

#### Bootstrapping everything

\_This method is slow\_

1. `yarn bootstrap --all`

2. Take a break ?

3. `yarn test` (to verify everything worked)

#### Building specific packages

If you're working on one or a few packages, for every change that you make, you have to rebuild those packages. To make the process easier, there is a CLI command for that:

- Run `yarn build` to bring you a list of packages to select from. There will be also an option to run in watch mode.

- Run `yarn build <package-name>` to build that package specifically. \

For the package name, use its short version. Example: for `@storybook/addon-docs`, run `yarn build addon-docs`.

- Run `yarn build --all` to build everything.

- Add `--watch` to run automatically in watch more if you are either building a selection of packages by name or building all.

Example: `yarn build core addon-docs --watch` or `yarn build --all --watch`.

### Working with the kitchen sink apps

Within the `examples` folder of the Storybook repo, you will find kitchen sink examples of storybook implementations for the various platforms that storybook supports.

Not only do these show many of the options and add-ons available, they are also automatically linked to all the development packages. We highly encourage you to use these to develop/test contributions on.

#### React and Vue

1. `cd examples/official-storybook`

2. `yarn storybook`

3. Verify that your local version works

### Working with your own app

#### Linking Storybook

Storybook is broken up into sub-projects that you can install as you need them. For this example, we will be working with `@storybook/react`.

\*\*Note:\*\* You need to `yarn link` from inside the subproject you are working on \*\*\_NOT\_\*\* the storybook root directory

1. `cd app/react`

2. `yarn link`

#### Connecting Your App To Storybook

\*\*\_Note:\_\*\* If you aren't seeing addons after linking storybook, you probably have a versioning issue which can be fixed by linking each addon you want to use.

This applies for the kitchen sink apps as well as your own projects.

\_Make sure `yarn dev` is running\_

##### 1. Setup storybook in your project

First we are going to install storybook, then we are going to link `@storybook/react` into our project. This will replace `node\_modules/@storybook/react` with a symlink to our local version of storybook.

1. `getstorybook`

2. `yarn storybook`

3. Verify that your local version works

##### 2. Link

\*\*\_Note\_\*\*: This process is the same for `@storybook/vue`, `@storybook/addon-foo`, etc

1. Go to your storybook \_root\_ directory

2. `yarn dev`

3. Wait until the output stops (changes you make will be transpiled into dist and logged here)

4. Go to your storybook-sandbox-app directory

5. `yarn link @storybook/react`

6. `yarn storybook`

#### Verify your local version is working

You should now have a working storybook dev environment up and running.

Save and go to `http://localhost:9011` (or wherever storybook is running)

If you don't see the changes rerun `yarn storybook` again in your sandbox app

## Release Guide

This section is for Storybook maintainers who will be creating releases. It assumes:

- yarn >= 1.3.2

- you've yarn linked `pr-log` from <https://github.com/storybookjs/pr-log/pull/2>

The current manual release sequence is as follows:

- Generate a changelog and verify the release by hand

- Push the changelog to master or the release branch

- Clean, build and publish the release

- Cut and paste the changelog to the github release page, and mark it as a (pre-) release

\*\*NOTE:\*\* The very first time you publish a scoped package (`@storybook/x`) you need to make sure that it's package.json contains the following

```js

"publishConfig": {

"access": "public"

}

```

This sequence applies to both releases and pre-releases, but differs slightly between the two.

\*\*NOTE: This is a work in progress. Don't try this unless you know what you're doing. We hope to automate this in CI, so this process is designed with that in mind.\*\*

#### Prerelease:

```sh

# make sure you current with origin/next.

git checkout next

git status

# generate changelog and edit as appropriate

# generates a Next section

yarn changelog:next x.y.z-alpha.a

# Edit the changelog/PRs as needed, then commit

git commit -m "x.y.z-alpha.a changelog"

# clean build

yarn bootstrap --reset --core

# publish and tag the release

yarn run publish:next

# update the release page

open https://github.com/storybookjs/storybook/releases

```

#### Full release:

```sh

# make sure you current with origin/master.

git checkout master

git status

# generate changelog and edit as appropriate

# generates a vNext section

yarn changelog x.y.z

# Edit the changelog/PRs as needed, then commit

git commit -m "x.y.z changelog"

# clean build

yarn bootstrap --reset --core

# publish and tag the release

yarn run publish:latest

# update the release page

open https://github.com/storybookjs/storybook/releases

```