---

category: Introduction

page: Contributing

---

# Contributing to baseui

## Getting started

1. Clone the repo locally and run `yarn` to install dependencies from npm.

```bash

git clone git@github.com:uber/baseweb.git

cd baseweb

yarn

```

2. To start a website to develop against, run `yarn documentation:dev`.

3. To unit test your changes run `yarn test` or `yarn test --watch` to continuously run the relevant tests.

Do you miss a component? Would you like to extend the featureset of a component?

This document helps you navigate the process.

## Contributions we won't accept

While we are extremely grateful for all the contributions we get, sometimes we have to say no to some pull requests.

Usually, we reject contributions if they meet any of the following requirements:

- Introduces a utility function/component, that's not used by Base Web itself.

- New components that were not approved before sending the pull request. To make sure you don't run into any issues landing your new component to the library, please open a GitHub issue first to discuss the new addition.

- Slight alterations of existing components - like introducing a new component called "Fancy Button".

- Breaking changes - if your changeset introduces API changes, please make sure to do them in a backward-compatible way.

- The PR includes opinionated changes that are not necessary - examples for this include introducing destructuring or moving files around based on personal preference.

- Most examples for the documentation site that shows integration with a third-party library or service - we welcome these guides as blogposts. If you are interested in contributing one, please open a GitHub issue with the proposal!

## Definition of done

- Each component has a drop-in, stateful, stateless, styled (presentation) components exported

- When you add examples for the documentation site, try to prioritize stateless examples with hooks

- Browser accessibility support and aria attributes

- A11y rules can be found [here](https://dequeuniversity.com/rules/axe/3.0/),

- Run `yarn lint` , which has eslint-plugin-jsx-a11y running as part of it https://github.com/evcohen/eslint-plugin-jsx-a11y#supported-rules,

- Check out `TESTING.md` to learn how to run end-to-end tests

- [Styletron](https://www.styletron.org/) for CSS-in-JS styling

- Unit tests with [jest](https://jestjs.io/en/) and [enzyme](https://airbnb.io/enzyme/)

- [Flow](https://flow.org/) type coverage for all relevant component code and tests

- TypeScript coverage for the API

- Documentation added to the docs site. You start the doc site using `yarn documentation:dev:watch`.

## Git Commit Formatting

Commit messages should be formatted according to [commitlint](https://commitlint.js.org/#/concepts-commit-conventions) specifications. Doing so allows us to better document the baseweb changelog.

## Sending Pull Requests

When send a pull request, please make sure that you have one of the [following labels](https://github.com/uber-workflow/probot-app-pr-label/blob/master/index.js#L20) set:

- breaking

- feature

- bugfix

- docs

- discussion

- release

- prerelease

- greenkeeping

## Environment Variables

We use a number of environment variables for our build process. Anything used by Buildkite should be stored with our secrets (ask a team member if you need access) and forwarded to the appropriate service in `docker-compose.yml`. Anything needed to build the documentation site should be added to Vercel.

For local development the only environment variables you may need to set up are for the documentation site's Figma based pages (/guidelines). If you are working on code for those pages you will want to create a `.env` file locally and populate `FIGMA\_AUTH\_TOKEN` as well as `FIGMA\_FILE\_ID`. You can then use `yarn documentation:dev:watch:env` to automatically load those variables in development.