# Contributing to pdfkit

## Table of Contents

- [Contributing to pdfkit](#contributing-to-pdfkit)

- [Table of Contents](#table-of-contents)

- [Code Organization](#code-organization)

- [Setting Up the project locally](#setting-up-the-project-locally)

- [Running and writing tests](#running-and-writing-tests)

- [Submitting a Pull Request](#submitting-a-pull-request)

## Code Organization

pdfkit is organized in the following folders:

- `lib`: The actual source code.

- `js`: The built / distributable code.

- `docs`: Code and artifacts to generate documentation.

- `demo`: Node and browser demos.

- `tests/unit`: Tests behavior of specific classes / methods.

- `tests/integration`: Compare the pdf output against a reference.

\*\*Working on your first Pull Request?\*\* You can learn how from this \_free\_ series [How to Contribute to an Open Source Project on GitHub](https://egghead.io/series/how-to-contribute-to-an-open-source-project-on-github)

## Setting Up the project locally

To install the project you need to have `node`

1. [Fork](https://help.github.com/articles/fork-a-repo/) the project, clone your fork:

```

# Clone your fork

git clone https://github.com/<your-username>/pdfkit.git

# Navigate to the newly cloned directory

cd pdfkit

```

2. `npm install` to install dependencies

3. `npm run build` to build the library

4. `npm run demo` to run the demo (check demo/out.pdf)

5. `npm run browser-demo` to run the browser demo (check demo/browser.html)

> Tip: Keep your `master` branch pointing at the original repository and make

> pull requests from branches on your fork. To do this, run:

>

> ```

> git remote add upstream https://github.com/foliojs/pdfkit.git

> git fetch upstream

> git branch --set-upstream-to=upstream/master master

> ```

>

> This will add the original repository as a "remote" called "upstream,"

> then fetch the git information from that remote, then set your local `master`

> branch to use the upstream master branch whenever you run `git pull`.

> Then you can make all of your pull request branches based on this `master`

> branch. Whenever you want to update your version of `master`, do a regular

> `git pull`.

## Running and writing tests

Tests are run using [Jest](http://jestjs.io/) and are categorized as integration and unit tests.

Integration tests check the pdf output against a reference stored as snapshots. While is served well to avoid regressions it has some disadvantages like small (correct) changes requiring to update all snapshots

Unit tests checks behavior os specific classes / methods isolatedly. It covers relatively small portion of code but is preferred way of writing new tests going forward

Tests commands

\* `npm run test`: Run all tests

\* `npm run test:unit`: Run unit tests

\* `npm run test:integration`: Run integration tests

To write new tests, look for the \*.spec.js files at `test/unit` and `test/integration` as examples

## Submitting a Pull Request

Please go through existing issues and pull requests to check if somebody else is already working on it.

Also, make sure to run the tests and lint the code before you commit your changes.

\*\*Preferentially, tests should be added to check the changed behavior even if is a bug fix. Unit tests are preferred over integration ones\*\*