Code Contributions

The beauty of contributing to open source is that you can clone your favorite project, get it running locally, and test out experiments and changes in real time! You get the opportunity to participate in a community while feeling a wizard doing it. For instructions on contributing to the docs, visit the docs contributions page.

Gatsby uses a "monorepo" pattern to manage its many dependencies and relies on Lerna and Yarn to configure the repository for both active development and documentation infrastructure changes.

Yarn is a package manager for your code, similar to npm. While npm is used to develop Gatsby sites with the CLI, contributing to the Gatsby repo requires Yarn for the following reason: we use Yarn's workspaces feature that comes really handy for monorepos. It allows us to install dependencies from multiple `package.json` files in sub-folders, enabling a faster and lighter installation process.

Setting up your local dev environment

Install Node and Yarn

- Ensure you have the latest **LTS** version of Node installed (`>= 14.15.0`) by executing `node --version`
- Install the Yarn package manager.
- Ensure you have the version 1 of Yarn installed (`^1`) by executing `yarn --version`. The Gatsby monorepo does not yet support later versions of Yarn.

Fork, clone, and branch the repository

- Fork the official `gatsbyjs/gatsby` repository.
- Clone your fork: `git clone https://github.com/<your-username>/gatsby.git`
- Set up repo and install dependencies: `yarn run bootstrap`
- Create a topic branch: `git checkout -b topics/new-feature-name`
- Run `yarn run watch` from the root of the repo to watch for changes to packages' source code and compile these changes on-the-fly as you work.

  - Note that the watch command can be resource intensive. To limit it to the packages you're working on, add a scope flag, like `yarn run watch --scope={gatsby,gatsby-cli}`.
  - To watch just one package such as `gatsby`, run `yarn run watch --scope=gatsby`.

  Note: Optionally you can run `git clone --depth=1 https://github.com/<your-username>/gatsby.` to do a shallow clone (smaller download size) rather than a deep clone, however this sometimes leads to problems when you want to reference older upstream branches.

Testing out changes in an example project

- Install gatsby-dev-cli:
  - Make sure you have the Gatsby Dev CLI installed with `gatsby-dev -v`
  - If not, install globally: `yarn global add gatsby-dev-cli`
  - Run `gatsby-dev --set-path-to-repo /path/to/my/forked/version/gatsby` to point `gatsby-dev-cli` to your fork
- Run `yarn install` in each of the sites you're testing.
- For each of your Gatsby test sites, run the `gatsby-dev` command inside the test site's directory to copy the built files from your cloned copy of Gatsby. It'll watch for your changes to Gatsby packages and copy them into the site. For more detailed instructions see the gatsby-dev-cli README and check out the gatsby-dev-cli demo video.
  - To copy files from just one package such as `gatsby`, run `gatsby-dev --packages gatsby`
  - Note: If you plan to modify packages that are exported from `gatsby` directly, you need to either add those manually to your test sites so that they are listed in `package.json` (e.g. `yarn add gatsby-link`), or specify them explicitly with `gatsby-dev --packages gatsby-link`).
- If you've recently run `gatsby-dev` your `node_modules` will be out of sync with current published packages. In order to undo this, you can remove the `node_modules` directory, revert any changes to `package.json` and lockfiles, and reinstall modules with `npm install` or `yarn install` . Alternatively, you can run:

  ```
  git checkout package.json; yarn --force
  ```

Add tests

- Add tests and code for your changes.
  - Begin by adding unit tests with Jest.
  - If you want to test out this feature in a more real-world application you can also consider adding integration tests and end-to-end tests.

  - If you're unsure on how to add tests or which types of test, ask in a comment in the pull request
- Once you're done, make sure all unit tests still pass: `yarn test`.
  - To run tests for a single package you can run: `yarn jest <package-name>`.
  - To run a single test file you can run: `yarn jest <file-path>`.
  - Once you push your changes to GitHub, the CI will run tests in a controlled environment and might uncover failures on e.g. different type of machines (Linux vs. Windows)

If you're adding e2e tests and want to run them against local changes:

- In the root of the monorepo, run `yarn lerna run build --scope=<package-name>` where `package-name` is the directory containing the changes you're testing.
- Run `gatsby-dev` inside your specific e2e test directory, for example `e2e-tests/themes/development-runtime`.
- While the previous step is running, open a new terminal window and run `yarn test` in that same e2e test directory.

Troubleshooting

At any point during the contributing process the Gatsby team would love to help! For help with a specific problem you can open an Discussion on GitHub. Or drop in to our Discord server for general community discussion and support.

- When you went through the initial setup some time ago and now want to contribute something new, you should make sure to sync your fork with the latest changes from the primary branch on gatsbyjs/gatsby. Otherwise, you might run into issues where files are not found as they were renamed, moved, or deleted.
- After syncing your fork, run `yarn run bootstrap` to compile all packages. When files or tests depend on the build output (files in `/dist` directories) they might fail otherwise.
- Make sure to run `yarn run watch` on the packages' source code you're changing.

How to open a pull request

If you've made all your changes, added tests, and want to contribute your changes to Gatsby, you can head over to the How to open a pull request documentation to learn more.

Other contributions

Creating your own plugins

If you create a loader or plugin, we would love for you to open source it and put it on npm. For more information on creating custom plugins, please see the

documentation for plugins and the API specification.

Contributing example sites

Gatsby's policy is that "Using" example sites (like those in the examples part of the repo) should only be around plugins that are maintained by the core team as it's hard to keep things up to date otherwise.

To contribute example sites, it is recommended to create your own GitHub repo and link to it from your source plugin, etc. You can also create a starter project and submit it to the starter showcase.

Debugging the build process

Check Debugging the build process page to learn how to debug Gatsby.

Feedback

At any point during the contributing process the Gatsby team would love to help! For help with a specific problem you can open an Discussion on GitHub. Or drop in to our Discord server for general community discussion and support.