# Contributing to Calypso

Hi! Thank you for your interest in contributing to Calypso, we really appreciate it.

There are many ways to contribute – reporting bugs, feature suggestions, fixing bugs, submitting pull requests for enhancements.

With over 10,000 PRs we have merged all sorts of different contributions from within Automattic and from the community, all working together to make Calypso a better experience for everyone.

## Reporting Bugs, Asking Questions, Sending Suggestions

Just [file a GitHub issue](https://github.com/Automattic/wp-calypso/issues/), that’s all. If you want to prefix the title with a “Question:”, “Bug:”, or the general area of the application, that would be helpful, but it is by no means mandatory. If you have write access, add the appropriate labels.

If you’re filing a bug, specific steps to reproduce are helpful. Please include the URL of the page that has the bug, along with what you expected to see and what happened instead.

Here is a [handy link for submitting a new bug](https://github.com/Automattic/wp-calypso/issues/new?labels%5B%5D=%5BType%5D%20Bug).

Feel free to share your unique context to help us understand your perspective. You can add context tags such as: `#journey` `#anecdote` `#narrative` `#context` `#empathy` `#perspective` `#reallife` `#dogfooding` `#livesharing` `#flowsharing` `#anxiety` `#anxiety-flow` `#stresscase` `#painpoint`. We'd also love to know how you found the bug: `#dogfooding`, `#manual-testing`, `#automated-testing`, or `#user-report` if applicable.

## Installing Calypso Locally

If you’d like to contribute code, first, you will need to run Calypso locally. Here is the short version:

1. Make sure you have `git`, `node`, and `npm` installed.

2. Clone this repository locally.

3. Add `127.0.0.1 calypso.localhost` to your local hosts file.

4. Execute `yarn start` from the root directory of the repository.

5. Open `http://calypso.localhost:3000` in your browser.

For more detailed instructions, see [Installing Calypso](../docs/install.md).

## Development Workflow

Running `yarn start` will build all the code and continuously watch the front-end JS and CSS/Sass for changes and rebuild accordingly.

See [Development Workflow](../docs/development-workflow.md) for details on how to run tests, control what debug messages to receive, and where to look for errors and warnings.

## Pull Requests

### Code Reviews

Code reviews are an important part of the Calypso workflow. They help to keep code quality consistent, and they help every person working on Calypso learn and improve over time. We want to make you the best Calypso contributor you can be.

Every PR should be reviewed and approved by someone other than the author, even if the author has write access. Fresh eyes can find problems that can hide in the open if you’ve been working on the code for a while.

The recommended way of finding an appropriate person to review your code is by [blaming](https://help.github.com/articles/using-git-blame-to-trace-changes-in-a-file/) one of the files you are updating and looking at who was responsible for previous commits on that file.

Then, you may ask that person to review your code by mentioning their GitHub username on the PR comments like this:

```

cc @username

```

\*Everyone\* is encouraged to review PRs and add feedback and ask questions, even people who are new to Calypso. Also, don’t just review PRs about what you’re working on. Reading other people’s code is a great way to learn new techniques, and seeing code outside of your own feature helps you to see patterns across the project. It’s also helpful to see the feedback other contributors are getting on their PRs.

### Coding Standards & Guidelines

Consistent coding style makes the code so much easier to read. Here are ours:

\* [All Coding Guidelines](../docs/coding-guidelines.md)

- [JavaScript](../docs/coding-guidelines/javascript.md)

- [TypeScript](../docs/coding-guidelines/typescript.md)

- [CSS/Sass](../docs/coding-guidelines/css.md)

- [HTML](../docs/coding-guidelines/html.md)

- [React Components](../docs/components.md)

\* [I18n Guidelines](https://github.com/Automattic/i18n-calypso/blob/HEAD/README.md)

\* [A11y Checklist](../docs/accessibility-checklist.md)

### Lifecycle of a Pull Request

When you’re first starting out, your natural instinct when creating a new feature will be to create a local feature branch, and start building away. If you start doing this, \*stop\*, take your hands off the keyboard, grab a coffee and read on. :)

\*\*It’s important to break your feature down into small pieces first\*\*, each piece should become its own pull request. Even if after finishing the first piece your feature isn’t functional, that is okay, we love merging unfinished code early and often. You can place your feature behind a [feature-check](../config/README.md#feature-flags) to make sure it’s not exposed until all pieces are completed.

Once you know what the first small piece of your feature will be, follow this general process while working:

1. [Fork](https://help.github.com/articles/fork-a-repo/) the project and create a new branch, using [the branch naming scheme](../docs/git-workflow.md#branch-naming-scheme), \_e.g.\_ `add/video-preview` or `fix/1337-language-too-geeky`.

2. Make your first commit: any will do even if empty or trivial, but we need something in order to create the initial pull request. [Create the pull request](https://help.github.com/articles/creating-a-pull-request-from-a-fork/) and prefix the name with the section of the product, \_e.g.\_ \_Posts: Prepare store for desktop app\_. Don’t worry too much if there’s no obvious prefix.

- Write a detailed description of the problem you are solving, the part of Calypso it affects, and how you plan on going about solving it.

- If you have write access, add the \*\*<span class="label status-in-progress">[Status] In Progress</span>\*\* label or wait until somebody adds it. This indicates that the pull request isn’t ready for a review and may still be incomplete. On the other hand, it welcomes early feedback and encourages collaboration during the development process.

3. Start developing and pushing out commits to your new branch.

- You can use a branch prefix like `add/`, `update/`, `try/` or `fix/` that represents the type of work you're doing. Avoid using the `renovate/` prefix in your branch name (it is used by the [Renovate Bot](https://renovatebot.com/) that automatically updates our NPM dependencies and your `renovate/\*` branch will confuse the bot).

- Push your changes out frequently and try to avoid getting stuck in a long-running branch or a merge nightmare. Smaller changes are much easier to review and to deal with potential conflicts.

- Follow the [merge checklist](../docs/merge-checklist.md) before pushing. This ensures that your code follows the style guidelines and doesn’t accidentally introduce any errors or regressions.

- Note that you can automate some of these tasks by setting up [githooks](../docs/coding-guidelines/javascript.md#setting-up-githooks) and they will run whenever you `git commit`.

- Don’t be afraid to change, [squash](http://gitready.com/advanced/2009/02/10/squashing-commits-with-rebase.html), and rearrange commits or to force push - `git push -f origin fix/something-broken`. Keep in mind, however, that if other people are committing on the same branch then you can mess up their history. You are perfectly safe if you are the only one pushing commits to that branch.

- Squash minor commits such as typo fixes or [fixes to previous commits](http://fle.github.io/git-tip-keep-your-branch-clean-with-fixup-and-autosquash.html) in the pull request.

4. If you end up needing more than a few commits, consider splitting the pull request into separate components. Discuss in the new pull request and in the comments why the branch was broken apart and any changes that may have taken place that necessitated the split. Our goal is to catch early in the review process those pull requests that attempt to do too much.

5. When you feel that you are ready for a formal review or for merging into `master` make sure you check this list and our [merge checklist](../docs/merge-checklist.md).

- Make sure your branch merges cleanly and consider rebasing against `master` to keep the branch history short and clean.

- If there are visual changes, add before and after screenshots in the pull request comments.

- Add unit tests, or at a minimum, provide helpful instructions for the reviewer so they can test your changes. This will help speed up the review process.

- Ensure that your commit messages are [meaningful](http://robots.thoughtbot.com/5-useful-tips-for-a-better-commit-message).

6. Mention that the PR is ready for review or if you have write access remove the \*\*<span class="label status-in-progress">[Status] In Progress</span>\*\* label from the pull request and add the \*\*<span class="label status-needs-review">[Status] Needs Review</span>\*\* label - someone will provide feedback on the latest unreviewed changes. The reviewer will also mark the pull request as \*\*<span class="label status-needs-author-reply">[Status] Needs Author Reply</span>\*\* if they think you need to change anything.

7. If you get a ?, ?, ?, <img src="https://github.githubassets.com/images/icons/emoji/shipit.png" class="emoji" />, or a LGTM and the status has been changed to \*\*<span class="label status-ready-to-merge">[Status] Ready to Merge</span>\*\* – this is great – the pull request is ready to be merged into `master`.

Whether somebody is reviewing your code or you are reviewing somebody else’s code, [a positive mindset towards code reviews](https://medium.com/medium-eng/the-code-review-mindset-3280a4af0a89) helps a ton. We’re building something together that is greater than the sum of its parts.

If you feel yourself waiting for someone to review a PR, don’t hesitate to personally ask for someone to review it or to mention them on GitHub. \_The PR author is responsible for pushing the change through.\_

### Adding A New Component

If you'd like to add a new component to Calypso, please review our [new component checklist](../docs/new-component-checklist.md).

### We’re Here To Help

We encourage you to ask for help at any point. We want your first experience with Calypso to be a good one, so don’t be shy. If you’re wondering why something is the way it is, or how a decision was made, you can tag issues with \*\*<span class="label type-question">[Type] Question</span>\*\* or prefix them with “Question:”.

## License

Calypso is licensed under [GNU General Public License v2 (or later)](../LICENSE.md).

All materials contributed should be compatible with the GPLv2. This means that if you own the material, you agree to license it under the GPLv2 license. If you are contributing code that is not your own, such as adding a component from another Open Source project, or adding an `npm` package, you need to make sure you follow these steps:

1. Check that the code has a license. If you can't find one, you can try to contact the original author and get permission to use, or ask them to release under a compatible Open Source license.

2. Check the license is compatible with [GPLv2](http://www.gnu.org/licenses/license-list.en.html#GPLCompatibleLicenses), note that the Apache 2.0 license is \*not\* compatible.

3. Add the code source URL (e.g. a GitHub URL), the files where it's used in `wp-calypso` and the full license terms to [`CREDITS.md`](../CREDITS.md)

4. Add attribution to the code, if applicable. This line should include the copyright notice of the source, and a reference to the license contained in [`CREDITS.md`](../CREDITS.md)