Contributing to Meteor

We are excited to have your help building Meteor — both the platform and the community behind it. Please read the project overview and guidelines for contributing bug reports and new code, or it might be hard for the community to help you with your issue or pull request.

Project overview

Before we jump into detailed guidelines for opening and triaging issues and submitting pull requests, here is some information about how our project is structured and resources you should refer to as you start contributing.

Also, please take a few minutes to understand a few terms in Meteor context reading our Glossary.

Ways to contribute

There are many ways to contribute to the Meteor Project. Here's a list of technical contributions with increasing levels of involvement and required knowledge of Meteor's code and operations.

- Reporting a bug
- Triaging issues
- Contributing to documentation
- Finding work
- Submitting pull requests
- Reviewing pull requests
- Maintaining a community package

There are also several ways to contribute to the Meteor Project outside of GitHub, like organizing or speaking at Meetups and events and helping to moderate our forums.

If you can think of any changes to the project, documentation, or guide that would improve the contributor experience, let us know by opening an issue!

Finding work

Are you new here? Please check our issues `good-first-issue`.

We curate specific issues that would make great pull requests for community contributors by applying the `ready` label.

Any issue which does not have the `ready` label still requires discussion on implementation details but input and positive commentary is welcome! Any pull request opened on an issue which is not `confirmed` is still welcome, however the pull-request is more likely to be sent back for reworking than a `ready` issue.

If in doubt about the best way to implement something, please create additional conversation on the issue. You can also reach one of the core committers and they will help you to find something interesting to work on.

Project roles

Here are descriptions of the existing project roles, along with the current contributors taking on those roles today.

Reviewer

Reviwers are members of the community that help with Pull Requests reviews.

Current Reviewers:

- meteor
  - @filipenevola
  - @renanccastro
  - @StorytellerCZ
  - @zodern
  - @lorensr

Core Committer

The contributors with commit access to meteor/meteor are employees of Meteor Software Ltd or community members who have distinguished themselves in other contribution areas. If you want to become a core committer please start writing PRs.

Current Core Committers:

- @filipenevola
- @renanccastro
- @denihs

Developer Evangelist

- @filipenevola (Feel free to reach him out on Twitter)

Tracking project work

Right now, the best place to track the work being done on Meteor is to take a look at the latest release milestone here. Also, the Meteor Roadmap contains high-level information on the current priorities of the project.

Reporting a bug in Meteor

We welcome clear bug reports. If you've found a bug in Meteor that isn't a security risk, please file a report in our issue tracker. Before you file your issue, **search** to see if it has already been reported. If so, up-vote (using GitHub reactions) or add additional helpful details to the existing issue to show that it's affecting multiple people.

> *There is a separate procedure for security-related issues. If the issue you've found contains sensitive information or raises a security concern, email* `security@meteor.com` *instead, which will page the security team.*

A Meteor app has many moving parts, and it's often difficult to reproduce a bug based on just a few lines of code. So your report should include a link to a repository with a reproduction. By making it as easy as possible for others to reproduce your bug, you make it easier for your bug to be fixed.

It's likely that without a reproduction, contributors won't look into fixing your issue and it will end up being closed.

A single code snippet is *not* a reproduction and neither is an entire application.

A reproduction works like this:

- Create a new Meteor app that displays the bug with as little code as possible. Try to delete any code that is unrelated to the precise bug you're reporting, including extraneous Atmosphere packages. Ideally, try to use as few source files as possible so that it's easy to see the whole reproduction on one screen, rather than making a large number of small files, even if that's not how you'd choose to structure an app.
- Create a new GitHub repository with a name like `meteor-reactivity-bug` (or if you're adding a new reproduction recipe to an existing issue, `meteor-issue-321`) and push your code to it. (Make sure to include the `.meteor/packages` and `.meteor/release` files!)

- Reproduce the bug from scratch, starting with a `git clone` command. Copy and paste the entire command-line input and output, starting with the `git clone` command, into the issue description of a new GitHub issue. Also describe any web browser interaction you need to do.
- If you reproduced the issue using a checkout of Meteor instead of using a released version that was pinned with a `.meteor/release` file, specify what commit in the Meteor repository was checked out.
- Mention what operating system you're using and what browser (if any).

If you can't provide a reproduction make this very clear in the issue and explain why that is the case.

If you want to submit a pull request that fixes your bug, that's even better. We love getting bugfix pull requests. Just make sure they're written with the correct style and *come with tests*. Read further down for more details on proposing changes to core code.

Feature requests

Feature requests are tracked in the Discussions.

Meteor is a big project with many sub-projects. Community is welcome to help in all the sub-projects. We use our roadmap to communicate the high-level features we're currently prioritizing.

Every additional feature adds a maintenance cost in addition to its value. This cost starts with the work of writing the feature or reviewing a community pull request. In addition to the core code change, attention needs to be paid to documentation, tests, maintainability, how the feature interacts with existing and speculative Meteor features, cross-browser/platform support, user experience/API considerations, etc. Once the feature is shipped, it then becomes the community's responsibility to fix future bugs related to the feature. In case the original author disappears, it's important that the feature has good tests and is widely used in order to be maintainable by other contributors.

For these reasons, we strongly encourage features to be implemented as Atmosphere or npm packages rather than changes to core. Try to re-work your feature request as a minimal set of hooks to core that enable the feature to be implemented as a package.

Feature requests should be well specified and unambiguous to have the greatest chance of being worked on by a contributor.

Finally, you can show your support for (or against!) features by up voting a discussion or by adding meaningful details which help the feature definition become more clear. Please do not comment with "+1" since it creates a lot of noise (e-mails, notifications, etc.) or comments unrelated with the feature definition.

Triaging issues

A great way to contribute to Meteor is by helping keep the issues in the repository clean and well organized. This process is called 'issue triage' and the steps are described here.

Learn how we use GitHub labels here

Documentation

If you'd like to contribute to Meteor's documentation, head over to https://docs.meteor.com or https://guide.meteor.com and if you find something that could be better click in "Edit on GitHub" footer to edit and submit a PR.

Blaze

Blaze lives in its own repository with its own issue tracker and feature prioritization and is not tracked within Meteor core.

Making changes to Meteor core

Eventually you may want to change something in a core Meteor package, or in the `meteor` command line tool. These changes have the highest standards for API design, for the names of symbols, for documentation, and for the code itself.

It may take some study to get comfortable with Meteor's core architecture. Each core package is designed to stand separately. At the same time, all the parts of core fit together to make the distinctive Meteor development experience. Core APIs should be consistent between the client and the server (not always workable; we don't have fibers on the client or a DOM on the server). We prefer synchronous APIs wherever possible: you can use `Meteor.wrapAsync` on the server to wrap async APIs that take a callback.

Above all, we are concerned with two design requirements when evaluating any change to a core package:

1. Nothing in Meteor should harm the experience of a new Meteor developer. That can be a difficult standard to reach, because we're concerned with the entire experience of developing and deploying an application. For example, we work hard to make sure that the Meteor docs don't force new users to understand advanced concepts before they need them. And we think a great deal about making our APIs as intuitive as possible, so that you can figure out a lot of Meteor without first having a long reading session with the docs.
2. Nothing in Meteor should preclude an expert from doing what they want. The low-level DDP API maps closely to the DDP wire protocol, for example, so that when the need arises you can control exactly what data gets sent to a client. It's okay to write syntactic sugar that makes the easy stuff easy, but if your change harms the experience of an expert then we'll probably prefer a different approach.

We have found that writing software to meet both these standards at the same time is hard but incredibly rewarding. We hope you come to feel the same way.

Understanding the core

For more information about how to work with Meteor core, take a look at the Development document which explains many important details, including how to run from a checkout, run tests, and more.

Proposing your change

You'll have the best chance of getting a change into core if you can build consensus in the community for it or if it is listed in the roadmap. Start by creating a well specified Discussion here.

Help drive discussion and advocate for your feature on the Github ticket (and perhaps the forums). The higher the demand for the feature and the greater the clarity of it's specification will determine the likelihood of a core contributor prioritizing your feature by flagging it with the `ready` label.

Split features up into smaller, logically separate chunks. It is unlikely that large and complicated PRs will be merged.

Once your feature has been labelled with `ready`, leave a comment letting people know you're working on it and you can begin work on the code. We have the label `in-development` to track the items in progress.

Submitting pull requests

Once you've come up with a good design, go ahead and submit a pull request (PR). When submitting a PR, please follow these guidelines:

- Sign the CLA (the bot will ask you do to this in the PR).
- Base all your work off of the **devel** branch. The **devel** branch is where active development happens. **We do not merge pull requests directly into master.**
- Name your branch to match the feature/bug fix that you are submitting.
- Limit yourself to one feature or bug fix per pull request.
- Include tests that prove your code works.
- Follow appropriate style for code contributions and commit messages
- Bump the version of the changed package accordingly
    - If your changes are ok to be released without a whole new Meteor version bump just the patch, for example, 2.4.5 will become 2.4.6.
    - If your changes need a new Meteor version because they are affecting many parts or they depend on changes in the meteor-tool bump the minor, for example, 2.4.5 will become 2.5.0.
    - If your change is a major rewrite then bump the major, for example, 2.4.5 will become 3.0.0.
    - If you bump anything that is not the patch you will need to wait a new Meteor version to have your changes available. This is how Meteor core packages work.
- Be sure your author field in git is properly filled out with your full name and email address so we can credit you.
- You can submit PRs that are not ready yet, submit them as Draft on GitHub and explain what is left and also if you need help.