Reviewing a pull request can take a considerable amount of time. In some cases, the review might require more time to perform than it took someone to write and submit their changes! It's therefore necessary to do some preliminary work to ensure each pull request is in a good state to be reviewed.

A pull request should consist of three main sections:

- A summary. This helps us understand the motivation behind the changes.
- A changelog. This helps us write the release notes. It also serves as a brief summary of your changes.
- A test plan. This might be the most important part of your pull request. A test plan should be a reproducible step-by-step guide so that a reviewer can verify your change is working as intended. It's also a good idea to attach screenshots or videos for user visible changes.

Any one pull request may require a deeper understanding of some area of React Native that you may not be familiar with. Even if you don't feel like you are the right person to review a pull request, you may still help by adding labels or asking the author for more information.

Reviewing PRs

Pull Requests need to be reviewed and approved using GitHub's review feature before they can be merged. While anyone has the ability to review and approve a pull request, we typically only consider a pull request ready to be merged when the approval comes from one of the contributors.

So you've found a pull request that you feel confident reviewing. Please make use of the GitHub Review feature, and clearly and politely communicate any suggested changes.

Consider starting with pull requests that have been flagged as lacking a changelog or test plan.

- PRs that appear to lack a changelog - take a look and see if you can add the changelog yourself by editing the PR. After doing so, remove the "Missing Changelog” label.
- PRs that are missing a test plan - open the pull request and look for a test plan. If the test plan looks sufficient, remove the "Missing Test Plan” label. If there is no test plan, or it looks incomplete, add a comment politely asking the author to consider adding a test plan.

A pull request must pass all the tests before it can be merged. They run on every commit on master and pull request. A quick way to help us get pull requests ready for review is to search for pull requests that are failing the pre-commit tests and determine if they need to be revised. The failing test is usually listed near the bottom of the thread, under “Some checks were not successful.”

- Take a quick glance at the latest tests runs on master. Is master green? If so,
  - Does it look like the failure may be related to the changes in this pull request? Ask the author to investigate.
  - Even if master is currently green, consider the possibility that the commits in the pull requests may be based off a commit from a point in time when master was broken. If you believe this may be the case, ask the author to rebase their changes on top of master in order to pull in any fixes that may have landed after they started working on the pull request.
- If master appears to be broken, look for any issues labeled as “CI Test Failure”.
  - If you find an issue that seems related to the failure on master, go back to the pull request and thank the author for proposing these changes, and let them know that the test failure may be

    unrelated to their particular change (do not forget to link back to the CI Test Failure issue, as this will help the author know when they can try running tests again).
    - If you cannot find an existing CI Test Failure issue that describes the problem you've observed on master, please submit a new issue and use the “CI Test Failure” label to let others know that master is broken (see this issue for an example).

How we prioritize PRs

Members of the React Native team at Facebook aim to review pull requests quickly and most PRs will get a response within a week.

How does a PR get merged?

The React Native GitHub repository is actually a mirror of a subdirectory from one of Facebook's monorepos. Pull requests are therefore not merged in the traditional sense. Instead, they need to be imported into Facebook's internal code review system as a “diff”. Once imported, the changes will go through a suite of tests. Some of these tests are land-blocking, meaning they need to succeed before the contents of the diff can be merged. Facebook always runs React Native from master and some changes may require a Facebook employee to attach internal changes to your pull request before it can be merged. For example, if you rename a module name, all Facebook internal callsites have to be updated in the same change in order to merge it. If the diff lands successfully, the changes will eventually get synced back to GitHub by ShipIt as a single commit.

Facebook employees are using a custom browser extension for GitHub that can import a pull request in one of two ways: the pull request can be “landed to fbsource”, meaning it will be imported and the resulting diff will be approved automatically, and barring any failures, the changes will eventually sync back to master. A pull request may also be “imported to Phabricator”, meaning the changes will be copied to an internal diff that will require further review and approval before it can land.

Screenshot of the custom browser extension. Two buttons are visible: a green one that is titled "Import to Phabricator", and a grey one with "Land to fbsource" in red text.

Bots

As you review and work on pull requests, you might encounter comments left by a handful of GitHub bot accounts. These bots have been set up to aid in the pull request review process. See the Bots Reference to learn more.

Pull Request Labels

- `Merged` : Applied to a closed PR to indicate that its changes have been incorporated into the core repository. This label is necessary because pull requests are not merged directly on GitHub. Instead, a patch with the PR's changes is imported and queued up for code review. Once approved, the result of applying those changes on top of Facebook's internal monorepository gets synced out to GitHub as a new commit. GitHub does not attribute that commit back to the original PR, hence the need for a label that communicates the PR's true status.
- `Blocked on FB` : The PR has been imported, but the changes have not yet been applied.