Start out by looking at issues that need triage, as identified by the "Needs: Triage" label.

- Is this a request for code-level help with an individual app? Would this be a better fit for Stack Overflow? If so, apply the "Resolution: For Stack Overflow" label.
- Does this issue make appropriate use of the template? If not, apply the "Needs: Template" label.
- Does this issue mention the React Native version that was used? If not, apply the "Needs: Environment Info" label.
- Does this issue include a Snack, a code example, OR a list of steps to reproduce the issue? If not, apply the "Needs: Repro" label.

  We sometimes get issues that are quite obviously not appropriate for the GitHub issue tracker. Add the "Type: Invalid" label, and a bot will close the issue automatically.

Once you get to this point, you can transition to parsing the content of the issue itself. Does this issue include a **clear description** of the problem? If not, *politely* ask the issue author to update their issue with the necessary information, and apply a "Needs: Author Feedback" label. We aim to always be friendly and helpful and expect the same from every member of our community.

Improving an Issue

If the issue contains all the necessary information, take a moment to consider if the issue can still be improved in some way. Is the formatting alright? You may lightly edit the issue to improve readability as needed. If the issue contains an unformatted block of code, surround it with three back ticks (“‘) to convert it into a markdown code block. Are there any labels you can add to help categorize it better? If the issue only affects Android apps, you may add a "Platform: Android" label. Perhaps the issue only manifests itself when developing on Windows, in which case you might add the "Platform: Windows" label. We have a long list of labels, please take a look and see if anything might apply!

Handling Duplicates

As you work through these issues, you will start to get a better understanding of the type of problems that get reported. You may even start noticing the same issue gets reported. In these cases, you can close the issue and add a comment that says "Duplicate of #issue". By following this convention, GitHub will automatically mark the issue as a duplicate.

Assessing Impact

Next up, we need to determine how severe the issue is.

- *Is this a potential* ***release blocker****?* These issues should be addressed within the next week or two, as they may prevent release coordinators

from cutting a clean release candidate. Issues that might be labeled as such could be regressions that break one of our pre-commit tests. Avoid flagging an issue as a release blocker if it has been around for some time (if the issue is already present in one or more releases, it cannot be a RC blocker by definition).

- *Does this cause the app to **crash**?* These are issues that cause a React Native to crash unexpectedly. These may lead to a poor user experience if not caught early.
- *Is this a **bug**?* Describes something that is not working as expected. It'd be nice to get it fixed at some point, but it's not serious enough to block the release train. Even if the issue causes a crash, if there's a reasonable workaround available, it can be classified as a regular bug.
- *Is this a **good first issue**?* These are issues that do not require a deep understanding and familiarity with the repo. GitHub will surface these issues to folks interested in becoming contributors. Keep in mind that issues labeled this way may not end up getting fixed right away.