When editing Flutter code, it's important to check the code with the analyzer.

If your IDE supports doing this automatically (e.g. using Android Studio with the Flutter plugin), then that is the easiest solution.

Alternatively, you can use the `flutter analyze --flutter-repo` tool on the console. When using the console, you will want to manually run `flutter update-packages` each time you update your tree, so that the `pubspec.yaml` files are processed to obtain all the dependencies for every package in the repository. If you don't do this, you may get bogus error messages about core classes like `Offset` from `dart:ui`. This is because `flutter analyze` does not automatically attempt to update dependencies, since doing so can take a long time and is only necessary when the tree has been updated.

For a one-off, use `flutter analyze --flutter-repo`.

For continuous analysis, use `flutter analyze --flutter-repo --watch`.

If you omit the `--flutter-repo` option you may end up in a confusing state because that will assume you want to check a single package and the flutter repository has several packages.

If you want to see how many members are missing dartdocs, provide the additional argument `--dartdocs`.

You can use the `--write` argument with `--watch` to cause each update to write all the results to a file in ASCII. This can be used e.g. with Emacs' `compile` mode to quickly dump all the latest analysis results into the compilation buffer so that they are recognized as error messages that Emacs can jump to.