# Contribution Guidelines

Thank you for visiting the Defold open source contribution guidelines. We invite you to help make Defold an even better product! You don't even have to be a coder to contribute! You can contribute in the following ways:

\* Improve the documentation

\* Suggest new features

\* Fix bugs

\* Implement new features

Before you start contributing we would like to ask you to familiarise yourself with our contribution guidelines and our code of conduct. Following these guidelines will make the contribution process a lot smoother and faster.

Thanks! ?

/ The Defold Foundation and the Defold development team

## Code of conduct

We ask of all contributors, project members and users to follow our [Code of Conduct](CODE\_OF\_CONDUCT) to create an open and welcoming environment. We will remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to our Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that we deem inappropriate, threatening, offensive, or harmful.

## Documentation

The Defold manuals and examples are hosted in a [separate repository](https://github.com/defold/doc). Please pay it a visit if you wish to help improve our documentation! We need help to fix typos, update images and to write or improve tutorials and manuals.

Note that the API reference is generated from the source code of the Defold engine. If you wish to submit improvements to the API reference it should be done as Pull Requests (see below).

## Feature requests

We value your ideas on how to improve Defold! You can submit feature requests to our public issue tracker using the [Feature Request template](https://github.com/defold/defold/issues/new?assignees=&labels=feature+request&template=feature\_request.md&title=). Some guidelines:

\* Please [search already submitted feature requests](https://github.com/defold/defold/labels/feature\_request) before submitting your feature request. Maybe someone has already suggested the same thing or something only slightly different?

\* If there is an already submitted feature request we appreciate if you add additional information instead of creating a new feature request.

\* We will close any duplicate feature requests.

\* If you do submit a new Feature Request it is important that you fill in all fields marked as required.

\* We will not be able to consider a new Feature Request until all required fields are filled in.

\* Never submit more than one Feature Request in a single GitHub issue.

## Bug reports

Bugs should be reported to our public issue tracker using the [Bug Report template](https://github.com/defold/defold/issues/new?assignees=&labels=bug&template=bug\_report.md&title=). Some guidelines:

\* Please [search already reported bug reports](https://github.com/defold/defold/labels/bug) before reporting a bug.

\* If there is an already existing bug report we appreciate if you add additional information to the existing bug report instead of creating a new one.

\* We will close duplicate bug reports.

\* If you do create a new Bug Report it is important that you fill in all fields marked as required.

\* We will not be able to consider a new Bug Report until all required fields are filled in.

\* Never submit more than one Bug Report in a single GitHub issue.

## Pull requests

Submit Pull Requests (PRs) to contribute to the Defold engine and/or editor source code. PRs can introduce new features, fix bugs or improve on our API reference. PRs for new features and bug fixes must follow these guidelines:

\* Never submit a PR that does more than one thing.

\* Make sure to follow the [Best practices for code contributions](#best-practices-for-code-contributions).

\* You must sign our Contributor License Agreement (CLA) as part of submitting the PR.

\* The CLA ensures that you are entitled to contribute the code/documentation/translation to the project you’re contributing to and are willing to have it used in distributions and derivative works.

\* The process of digitally signing the CLA is integrated into the PR flow on GitHub.

\* Always reference existing issues covered by the PR.

\* Individual commits should have informative commit messages.

\* For new code or changes to existing code:

\* Should be covered by tests (unit, integration or system tests).

\* All tests must pass CI before a Pull Request is approved and can be merged.

\* Must follow the defined coding style.

\* Use the .clang\_format file for engine code.

\* Should if possible be tested on all target platforms (preferably on physical hardware).

\* Please note in the PR on which platforms the change has been tested.

\* All pull requests should be made to `defold:dev` by default. However, if your pull request \*only\* touches the editor and you want it to reach users as quickly as possible, it can be made towards the `defold:editor-dev` branch instead. Users will receive an editor update as soon as a change is merged into `defold:editor-dev`, so this should be done with care. The editor uses the latest stable release of the runtime, so any change that touches the runtime, the Bob command-line build tool, or makes a change to the file formats \*must\* be made towards the `defold:dev` branch, even if it contains mostly editor changes.

\* Pull requests should be given descriptive names: Issue-1234-Changed-the-flux-capacitor-to-use-uranium

\* Make sure to [link the issue to the pull request](https://help.github.com/en/github/managing-your-work-on-github/linking-a-pull-request-to-an-issue#linking-a-pull-request-to-an-issue-using-a-keyword)

### New features

If you want to add new functionality there are several additional steps that must be taken before you can submit a PR:

\* New functionality must be well designed and useful to not only your own project.

\* For game engine code there is always the option to create a native extension and share it via the Asset Portal if the functionality is specific to a certain type of project or in other ways not suitable for inclusion in the engine.

\* You should start by creating a [Feature Design ticket](https://github.com/defold/defold/issues/new?assignees=&labels=feature\_design&title=) to discuss the proposed solution.

\* The Feature Design should discuss the pros and cons of different solutions and it should document why a decision is made to implement the feature in a certain way.

\* Never submit a PR without first creating a design ticket.

\* Pull requests for new features without a corresponding GitHub issue will not be approved.

\* Reference any existing Feature Request covered by the design.

\* Any new API that is introduced should be well designed and future proof as we always try to avoid introducing breaking changes.

\* New engine functionality that breaks existing functionality should be avoided at all costs.

\* When the final design has been agreed upon the PR will be locked from further updates.

\* Reference the Feature Design in the PR.

### API reference

The API reference is automatically extracted from the source code as a separate build step. There are no additional considerations to be made for PRs to the documentation besides paying attention to the grammar and making sure that the changes correct and/or clarify existing documentation.

## Best practices for code contributions

When discussing bug fixes, new features or changes to existing features it is very important to think about both the problem and the solution.

You should always start by asking yourself:

\*"What is the problem that I am trying to solve?"\*

and follow up with:

\*"What is the least complex and most straight forward solution?"\*

Most contributions start with a Problem and result in a Solution. When discussing Problems and Solutions there are a few best practices to consider.

(These practices are inspired by the excellent ["Best practice for engine contributors"](https://docs.godotengine.org/en/latest/community/contributing/best\_practices\_for\_engine\_contributors.html) document in Godot)

### The Problem must comes first

Many developers enjoy the creative process of building systems and frameworks just for the sake of creating something. While this in itself can be both fun and rewarding it is something to pay attention to when working on a game engine. Every line of code you add will increase the build time, the complexity and the size of the code base, all of which are things you want to avoid in a game engine (or actually in all software).

Best practice: \*"Never add code that doesn't solve an actual problem."\*

When contributing to Defold we want the Problem to come first, not the Solution.

### Solutions should solve existing Problems

While it can sometimes be good to plan ahead and try to anticipate Problems down the road it can also lead to speculation and Solutions that are larger than they actually need to be.

Best practice: \*"Only solve problems that exist right now"\*

When contributing to Defold we want the Solution to solve a Problem that exists now.

### Solutions should solve frequent or complex Problems

The reason game developers use game engines instead of creating games completely from scratch is that game engines provide Solutions to complex and frequent Problems. This allows game developers to focus on creating games instead of solving technology problems. Game engines exist to solve Problems, but game engines can not solve all Problems. Some Problems require Solutions that have more to do with the game than with the engine and some Problems are easy to solve and can be left for the developer or a third party library to solve.

Best practice: \*"Do not try to solve all problems. Focus on frequent or complex problems."\*

When contributing to Defold we want to provide Solutions to Problems that game developers encounter frequently or Problems that are complex and hard to solve.

### Create one Solution per Problem

It may be tempting to try and come up with a clever Solution to solve many Problems at once. What usually happens is that these kinds of Solutions end up a lot more complex than Solutions that focus on solving a single Problem. It will also make it easier to reason about and review a change if it provides a Solution to a single Problem. Not to mention if the code needs to be refactored in the future!

Best practice: \*"A solution should solve a single problem"\*

When contributing to Defold we want each change to represent the Solution to a single Problem.

### Solutions should solve engine Problems

Game developers are faced with many different kinds of problems when developing games. Some are obviously game related, like how to best code the enemy behaviour or how to present the player inventory. Some Problems are not as obvious, like how to do pathfinding efficiently or how to create a scrolling list of UI elements. It is tempting to expect these kinds of Problems to be solved by the engine, and some game engines go down that route and provide Solutions to fairly high-level Problems. The design philosophy in Defold is to not solve these Problems but instead provide low-level building blocks that can be used to create Solutions for high-level Problems.

Best practice: \*"Solutions should provide low-level building blocks to solve high-level problems"\*

When contributing to Defold we want low-level Solutions that give game developers the functionality to solve high-level Problems on their own.