# Contributing

## Code

We do not have a formatting guideline \*yet\*, and expect you to write code with common sense in mind. Consistency is key, so look at the surrounding code and adhere to the same standards. Always check the diff before you commit (and more importantly: before filing a PR); many irrelevant changes usually mean your autoformat settings are off. Luckily, our project now includes a `.editorconfig` that automatically sets all the built-in formatting/language-rules. This is however only supported in Visual Studio 2017.

A large list of discussion items has been compiled [here](https://github.com/SubnauticaNitrox/Nitrox/issues/36), which hopefully make it into a guideline soon™.

### Formatting

- Use UpperCamelCase for methods, classes, properties and public fields.

- Use lowerCamelCase for local variables, private fields and function parameters.

- Remove unused `using` statements (Visual Studio has a shortcut to do this, as well as sort them).

- Put opening braces (`{`) on their own line.

- Encapsulate everything in braces, even single-line statements.

- Use `Optional<T>` for explicit nullable types.

- Use strict access modifiers where possible, and mark immutable variables wherever appropriate (`const`, `readonly` or `static readonly`, depending on the usecase).

- Remove `private set;` from properties if these are only assigned in the constructor.

- Assign simple data types (that do not require arguments) at declaration rather than in the constructor (such as `private readonly List<int> someList = new List<int>();`, instead of putting the `new` part in the constructor).

Again, consistency is key! If you're unsure, look at the rest of the code!

### Safe guidelines

When retrieving data from the game, use `Validate.NotNull` or similar where possible. Subnautica is still heavily under development, and things \*will\* change. Defensive coding standards help identify and fix these changes more easily.

To reduce redundant if sturctures, there are several helper functions that combine a certain retrieval function with a `Validation` check - it's recommended to use these. For example `GameObject.RequireComponent` and `GuidHelper.RequireObjectFrom`. In case of `Optional<T>` return types, these are unwrapped for you as well, saving the redundant checks \*and\* unwrapping.

### Help your fellow developers!

If code (especially patches that touch game internals) is/are not immediately clear, add a comment explaining the situation. This goes a long way, as you're basically presenting an overview of the research you did in the game code. And it helps others to figure out what's supposed to happen, in case the game changes and the patch is not functioning properly anymore.

### General OO(P) concepts

... such as "low coupling" and "high coherency", which basically means that you should try to reduce interaction between different pieces of code (classes) as much as possible (low coupling), and make sure that these pieces of code (classes) do the thing they are meant to do, and nothing else (high coherency). This goes a long way in code maintainability; less spaghetti code is always better :wink:

Also in this case, keep common sense in mind: if you're constantly accessing a lot of fields and methods of another class, consider tucking it away in a method in that class.

## Git workflow

### Git help

<sub><sup>Skip this section is if you are familiar with git.</sup></sub>

Git is there to help you develop and share code efficiently, though it may seem daunting at first. In order to use it effectively, we recommend you to read some tutorials so that you are familiar with the concept of commits, the way these are linked together, branches, and remotes.

When cloning a repository, git adds the clone url as a remote named `origin`. In this guide, we assume that's your fork. To keep your repository updated, add this official repo to your remotes:

```bash

# SSH:

git remote add upstream git@github.com:SubnauticaNitrox/Nitrox.git

# HTTPS:

git remote add upstream https://github.com/SubnauticaNitrox/Nitrox.git

```

We recommend to keep your master branch up to date with the offical master branch. This makes it easier to base feature branches on.

To go even further, you configure your master branch to pull from `SubnauticaNitrox`, and push to your own fork:

```bash

git config branch.master.remote upstream

git config branch.master.pushRemote origin

```

### Filing a PR

When filing a PR, we obviously expect the code to compile, run with no errors, and merge without conflicts.

To prevent these, and ensure that your code is compatible with the most recent 'version',

merge master into your branch, or rebase your branch on top of master. Even if git(hub) says your code can be merged without conflicts, there might be structural changes (renamings, moved files, refactors, etc), causing the final result to fail compilation, or break at runtime.

It is not desired to remove code just because "it doesn't work", or "causes exceptions in the log". If that's the case, try to fix it (recommended to file the changes in a separate PR), or notify the other Nitrox devs (by creating an issue on github, for example). All code is there for a reason - and someone else spent time creating it.