# Contributing

> First and foremost, thank you for stepping in and trying to be part of the effort.

### Goals

When expanding Kint, keep in mind its main priorities:

1. \*\*Ease of use\*\*: developers must be able to install Kint effortlessly and start using it immediately with no additional required steps. JS/CSS resources and the phar file should be rebuilt with `composer build` and committed when changed.

2. \*\*No feature creep\*\*: requests for edge use cases, especially ones that can be worked around with some configuration, should \*not\* be catered to. The definition of edge case is the popularity of the feature request, if it's popular - it's a workflow, if not - it might be an edge case.

3. \*\*Do not compromise information clarity\*\*: displaying relevant information is the entire purpose of Kint. Try to ensure your output is as clear as possible.

4. \*\*Code quality\*\*: the nature of Kint implies that it will be continued to develop for many years - at least to adapt to new language features and quirks. Keep the codebase as maintainable as possible. Keep code style and quality consistent by finishing every commit with `composer format`, and `composer analyze`.

5. \*\*Stable releases\*\*: Try not to push backwards compatibility breaking or untested changes to the master branch. A new branch will be created for upcoming major version upgrades.

6. \*\*Don't alter user input\*\*: For the parser to do proper recursion checking it has to take things by reference. Since it's taking things by reference there's a good chance your parser plugin can change your end-users data. \*\*NEVER ALTER INPUT VARIABLES\*\*

### Development instructions

```sh

composer install # Install dependencies

composer format # Format code style

composer analyze # Analyze code

composer build # Build JS/CSS resources, and the final phar file

```

Note: Full format and build depends on `npm` being in your `$PATH`.