# Contribution Guidelines

\*\*30 seconds of code\*\* is powered by the community, so feel free to contribute in any way you can to help us!

## Tools

Before you begin contributing, you should install the integration-tools globally on your machine:

```sh

npm install -g @30-seconds/integration-tools

```

This will allow you to use our customized tools for all of our content repositories.

## How you can help

- Submit pull requests with new snippets (see guidelines below) or snippet updates (tags, descriptions, explanations, typos, examples, code improvements).

- Open issues for things you want to see added, modified, discuss ideas or help out with existing issues.

## Ground rules

Breaking any of these rules will result in your pull request being closed. Please follow these guidelines above all else:

- \*\*Always be polite and respectful to others\*\* and try to follow the advice of the moderators/collaborators/owners.

- \*\*Only modify snippet files\*\*, never modify the generated files in the `snippet\_data` directory.

- \*\*Use the integration tools commands\*\* to generate new snippets, ensuring they have the correct name and are in the correct location.

- \*\*Follow snippet format exactly\*\*, otherwise your snippets will not be recognized correctly by the tools responsible for publishing them on the website. This includes such things as spacing and empty lines - if you accidentally make a mistake, consult the repository's [snippet template](snippet-template.md).

- \*\*Snippets should solve real-world problems\*\*, no matter how simple and should be abstract enough to be applied to different scenarios.

## Snippet creation

After installing the integration tools, you can run the following command:

```sh

create-new-snippet <my-snippet-name>

```

Replace `<my-snippet-name>` with the name of the snippet you are adding.

## Snippet guidelines

- Snippets must have all their frontmatter sections (title, tags etc.) filled.

- Snippet titles must correspond to the filename and follow the language and repository's naming conventions.

- Snippet tags must be comma-separated, contain a primary tag as seen on the website as their first tag and an expertise tag (`beginner`, `intermediate` or `advanced`) as their last tag.

- Snippet descriptions must be short and to the point. Explain \*what\* the snippet does and detail \*how\* the snippet works and the language features used in it.

- Snippet code and examples must be enclosed in appropriate, language-tagged blocks as shown in the snippet template, be short and use modern techniques and features. Also make sure to test your code before submitting.

- If your snippet contains arguments with default parameters, explain what happens if they are omitted when calling the function and what the default case is. Specify default parameters for arguments only if necessary.

- If your snippet uses recursion, use the `recursion` tag and explain the base cases.

- Try to strike a balance between readability, brevity, and performance.

- Always use soft tabs (2 spaces), never hard tabs.

- Leave a single space after a comma (`,`) character (both in the description and code).

- Define multiple variables on the same line, if possible. Use meaningful names (e.g. `letter` instead of `lt`) and follow existing conventions as seen in other snippets. Do not use trailing or leading underscores in variable names.