# Contributing

We are happy to accept new contributions. The earlier you share your ideas, the quicker we can collaborate.

Don't worry if you are stuck, or your pull request is incomplete or broken. We are here to help!

## Creating a new function

Details about creating a new function are listed [in the readme][].

You can visit [the functions wiki page][] for inspiration and examples.

See below for a common template that can be followed while creating a new function.

We love it when new functions:

\* are prefixed with `geometry\_`

\* are located in [geometry/functions/](/functions)

\* document their customization options [in the readme][]

\* share screenshots or screencasts in the pull request

## A sample function template

``` zsh

# Define defaults for relevant environment variables

: ${GEOMETRY\_<PLUGIN\_NAME>\_VAR1:="VALUE1"}

: ${GEOMETRY\_<PLUGIN\_NAME>\_VAR2:="VALUE2"}

# Define the function

geometry\_<function\_name>() {

# check if command exists for function to be useful

(( $+commands[command\_i\_need\_to\_work] )) || return

# Test that it makes sense to render this function

test -n "$SOME\_ENV\_THING\_FOR\_MY\_PLUGIN" || return

# Print something

command\_i\_need\_to\_work --version

}

```

## New function tutorial

If you want to set up your own custom function, it's pretty straightforward to do

so. All you need is to make sure it echos whatever you want printed to the prompt.

Let's assume you want to add a plugin that prints `(????)?` when the git branch

is clean and `(????)?????` when it's dirty.

Let's call it `my\_pretty\_git`.

```zsh

# my\_pretty\_git - show emoticons if in a git directory

```

It's good practice to check if it makes sense to display the plugin in the current context before echoing out anything.

```sh

my\_pretty\_git() {

(( $+commands[git] )) || return # return if `git` doesn't exist

[ -d $PWD/.git ] || return # return if we are not in a git repository

}

```

Now that we know we are in a useful context, let's setup some environment variables for customization:

```sh

my\_pretty\_git() {

(( $+commands[git] )) || return # return if `git` doesn't exist

[ -d $PWD/.git ] || return # return if we are not in a git repository

: ${MY\_PRETTY\_GIT\_CLEAN:="(????)?"}

: ${MY\_PRETTY\_GIT\_DIRTY:="(????)?????"}

}

```

Finally we can check the branch status and print accordingly:

```sh

my\_pretty\_git() {

(( $+commands[git] )) || return # return if `git` doesn't exist

[ -d $PWD/.git ] || return # return if we are not in a git repository

if [[ -z "$(git status --porcelain --ignore-submodules)" ]]; then

echo -n $MY\_PRETTY\_GIT\_CLEAN # make sure to use "-n" with your echo!

else

echo -n $MY\_PRETTY\_GIT\_DIRTY

fi

}

```

To help maintain an uncluttered prompt, functions should only render when in a

context where it makes sense. For example, the `geometry\_node` function

will only be display when in a npm or yarn-based project.

### Full working example

Save the following example as `my\_pretty\_git.zsh` anywhere.

Then just source the function and add it to `GEOMETRY\_PROMPT` or `GEOMETRY\_RPROMPT`.

```zsh

cat <<EOF > my\_pretty\_git.zsh

# my\_pretty\_git - show emoticons if in a git directory

my\_pretty\_git() {

(( $+commands[git] )) || return

[ -d $PWD/.git ] || return

: ${MY\_PRETTY\_GIT\_CLEAN:-"(????)?"}

: ${MY\_PRETTY\_GIT\_DIRTY:-"(????)?????"}

if [[ -z "$(git status --porcelain --ignore-submodules)" ]]; then

echo $MY\_PRETTY\_GIT\_CLEAN

else

echo $MY\_PRETTY\_GIT\_DIRTY

fi

}

EOF

source my\_pretty\_git.zsh

GEOMETRY\_RPROMPT+=(my\_pretty\_git)

```

[in the readme]: https://github.com/geometry-zsh/geometry/blob/master/readme.md

[the functions wiki page]: https://github.com/geometry-zsh/geometry/wiki/functions