# nvim-various-textobjs

Bundle of about a dozen custom text objects for Neovim. Includes text objects for: indentation, number, value, diagnostic, markdown link, double square bracket, and many more.



## List of all Text Objects

- `.indentation(startBorder, endBorder)`: Indentation text object. Similar to vim-indent-object, The two Boolean parameters determine whether the line in front is included (`aI` or `ai`). Setting both to `false` results in no border inclusion (`ii`).
- `.value(inner)`: Value of a key-value-pair, or the right-hand-side of a variable assignment. Looks for the first `:` or `=` in the line. Inner value excludes trailing comma or semicolon, outer value includes them. Always excludes trailing comments.*
- `.number(inner)`: Number text object. Inner number excludes decimal points and minus sign, outer number includes them.*
- `.diagnostic()`: Diagnostic from the built-in LSP. Similar to textobj-diagnostic.nvim.*
- `.subword()`: like `iw`, but treating dashes and underscores always as word delimiters, regardless of the `iskeyword` option.
- `.nearEoL()`: from cursor position to end of line minus 1 character. Useful to change everything except a trailing comma or semicolon.
- `.restOfParagraph()`: like `}`, but linewise.

**FileType specific** - `.mdlink(inner)`: Markdown link like `[title](url)`. Inner link only includes the link title inside the `[]`.* - `.mdFencedCodeBlock(inner)`: Markdown code block enclosed by three backticks. ``` - `.jsRegex(inner)`: JavaScript regex like `/exp/`. Inner regex excludes the surrounding `/`, outer regex includes them.* - `.cssSelector(inner)`: CSS class selector like `.my-class`. Similar to `iw`, but does not treat – as word-delimiter, and only accepts words with

leading `.` as selectors. Inner selector excludes the leading`.`, outer selector includes it.* - `.doubleSquareBrackets(inner)`: text surrounded by `[[` and `]]`. Multi-line strings in lua, conditionals in shell, or wikilinks in note-filetypes.*

> **Note**
> Text objects marked with ∗ seek up to 5 lines forward if the cursor is not standing on the text object. (Number of lines can be configured.)

## Installation and Setup

```
-- packer
use "chrisgrieser/nvim-various-textobjs"
```

A `.setup()` call is not required. It is only needed if you want to change the amount of lines below the cursor where the plugin looks for a text object:

```
require("various-textobjs").setup {
    -- default 5. Set to 0 to only look in the current line
    lookForwardLines = 10,
}
```

The plugin comes without any default keybindings. Set any keybindings you want to have yourself. All parameters are Boolean.

```
-- example: `an` for outer number, `in` for inner number
vim.keymap.set({"o", "x"}, "an", function () require("various-textobjs")
vim.keymap.set({"o", "x"}, "in", function () require("various-textobjs")
```

## Roadmap

- [ ] Figure out how to make dot-repeatability work. (Pointers are welcome.)

## Credits

Thanks to the Valuable Dev for their blogpost on how to get started with creating custom text objects.

## About me

In my day job, I am a sociologist studying the social mechanisms underlying the digital economy. For my PhD project, I investigate the governance of the app economy and how software ecosystems manage the tension between innovation and compatibility. If you are interested in this subject, feel free to get in touch.

**Profiles** - Discord - Academic Website - GitHub - Twitter - ResearchGate - LinkedIn