Tabstops

With tabstops you can make the editor cursor move inside a snippet. Use `$1` , `$2` to specify cursor locations. The number is the order in which tabstops will be visited, whereas `$0` denotes the final cursor position. Multiple tabstops are linked and updated in sync.

Placeholders

Placeholders are tabstops with values, like `${1:foo}` . The placeholder text will be inserted and selected such that it can be easily changed. Placeholders can nested, like `${1:another ${2:placeholder}}` .

Choice

Placeholders can have choices as values. The syntax is a comma-separated enumeration of values, enclosed with the pipe-character, e.g. `${1|one,two,three|}` . When inserted and selected choices will prompt the user to pick one of the values.

Variables

With `$name` or `${name:default}` you can insert the value of a variable. When a variable isn't set its *default* or the empty string is inserted. When a variable is unknown (that is, its name isn't defined) the name of the variable is inserted and it is transformed into a placeholder. The following variables can be used:

- `TM_SELECTED_TEXT` The currently selected text or the empty string
- `TM_CURRENT_LINE` The contents of the current line
- `TM_CURRENT_WORD` The contents of the word under cursor or the empty string
- `TM_LINE_INDEX` The zero-index based line number
- `TM_LINE_NUMBER` The one-index based line number
- `TM_FILENAME` The filename of the current document
- `TM_FILENAME_BASE` The filename of the current document without its extensions
- `TM_DIRECTORY` The directory of the current document
- `TM_FILEPATH` The full file path of the current document
- `RELATIVE_FILEPATH` The relative (to the opened workspace or folder) file path of the current document
- `CLIPBOARD` The contents of your clipboard
- `WORKSPACE_NAME` The name of the opened workspace or folder
- `WORKSPACE_FOLDER` The path of the opened workspace or folder

For inserting the current date and time:

- `CURRENT_YEAR` The current year
- `CURRENT_YEAR_SHORT` The current year's last two digits
- `CURRENT_MONTH` The month as two digits (example '02')
- `CURRENT_MONTH_NAME` The full name of the month (example 'July')
- `CURRENT_MONTH_NAME_SHORT` The short name of the month (example 'Jul')
- `CURRENT_DATE` The day of the month
- `CURRENT_DAY_NAME` The name of day (example 'Monday')
- `CURRENT_DAY_NAME_SHORT` The short name of the day (example 'Mon')
- `CURRENT_HOUR` The current hour in 24-hour clock format
- `CURRENT_MINUTE` The current minute

- `CURRENT_SECOND` The current second
- `CURRENT_SECONDS_UNIX` The number of seconds since the Unix epoch

For inserting random values:

- `RANDOM` 6 random Base-10 digits
- `RANDOM_HEX` 6 random Base-16 digits
- `UUID` A Version 4 UUID

Variable-Transform

Transformations allow to modify the value of a variable before it is being inserted. The definition of a transformation consists of three parts:

1. A regular expression that is matched against the value of a variable, or the empty string when the variable cannot be resolved.
2. A "format string" that allows to reference matching groups from the regular expression. The format string allows for conditional inserts and simple modifications.
3. Options that are passed to the regular expression

The following sample inserts the name of the current file without its ending, so from `foo.txt` it makes `foo`.

```
${TM_FILENAME/(.*)\..+$/$1/}
  |           |         | |
  |           |         | |-> no options
  |           |         |
  |           |         |-> references the contents of the first
  |           |             capture group
  |           |
  |           |-> regex to capture everything before
  |               the final `.suffix`
  |
  |-> resolves to the filename
```

Placeholder-Transform

Like a Variable-Transform, a transformation of a placeholder allows changing the inserted text for the placeholder when moving to the next tab stop. The inserted text is matched with the regular expression and the match or matches - depending on the options - are replaced with the specified replacement format text. Every occurrence of a placeholder can define its own transformation independently using the value of the first placeholder. The format for Placeholder-Transforms is the same as for Variable-Transforms.

The following sample removes an underscore at the beginning of the text. `_transform` becomes `transform`.

```
${1/^_(.*)/$1/}
  |   |    |  |-> No options
  |   |    |
  |   |    |-> Replace it with the first capture group
  |   |
  |   |-> Regular expression to capture everything after the underscore
  |
  |-> Placeholder Index
```

Grammar

Below is the EBNF for snippets. With `\` (backslash) you can escape `$`, `}` and `\`, within choice elements the backslash also escapes comma and pipe characters.

```
any         ::= tabstop | placeholder | choice | variable | text
tabstop     ::= '$' int
                | '${' int '}'
                | '${' int  transform '}'
placeholder ::= '${' int ':' any '}'
choice      ::= '${' int '|' text (',' text)* '|}'
variable    ::= '$' var | '${' var }'
                | '${' var ':' any '}'
                | '${' var transform '}'
transform   ::= '/' regex '/' (format | text)+ '/' options
format      ::= '$' int | '${' int '}'
                | '${' int ':' '/upcase' | '/downcase' | '/capitalize' | '/camelcase'
| '/pascalcase' '}'
                | '${' int ':+' if '}'
                | '${' int ':?' if ':' else '}'
                | '${' int ':-' else '}' | '${' int ':' else '}'
regex       ::= JavaScript Regular Expression value (ctor-string)
options     ::= JavaScript Regular Expression option (ctor-options)
var         ::= [_a-zA-Z] [_a-zA-Z0-9]*
int         ::= [0-9]+
text        ::= .*
```