Auto-closing parenthesis

2014/11/5 VimConf 2014
cohama / @c0hama

About me

cohama

- Twitter @c0hama
- from Nagoya
  - Nagoya.vim
- Agit.vim,
  the-ocamlspot.vim
- JavaScript, Haskell

Introduction

```
reverse(range(1, winnr('$'
```

How many) do we need to type?

Introduction

```
while (true) {
  if (p == 1) {
    foo();
    break;
```

How about } ?

Introduction

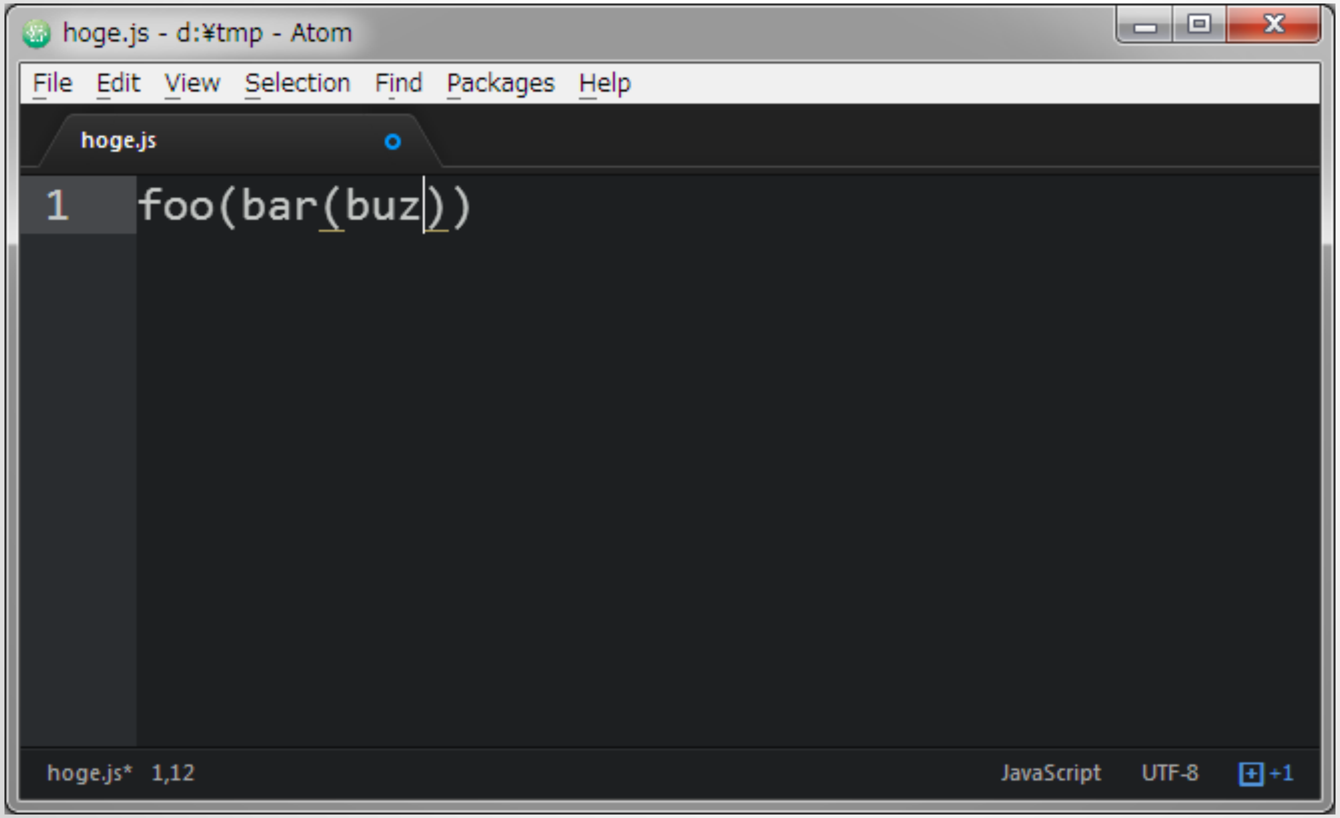
- We need auto-closing.

```
reverse(range(1, winnr('$')))
```

```
while (true) {
  if (p == 1) {
    foo();
    break;
  }
}
```

For example

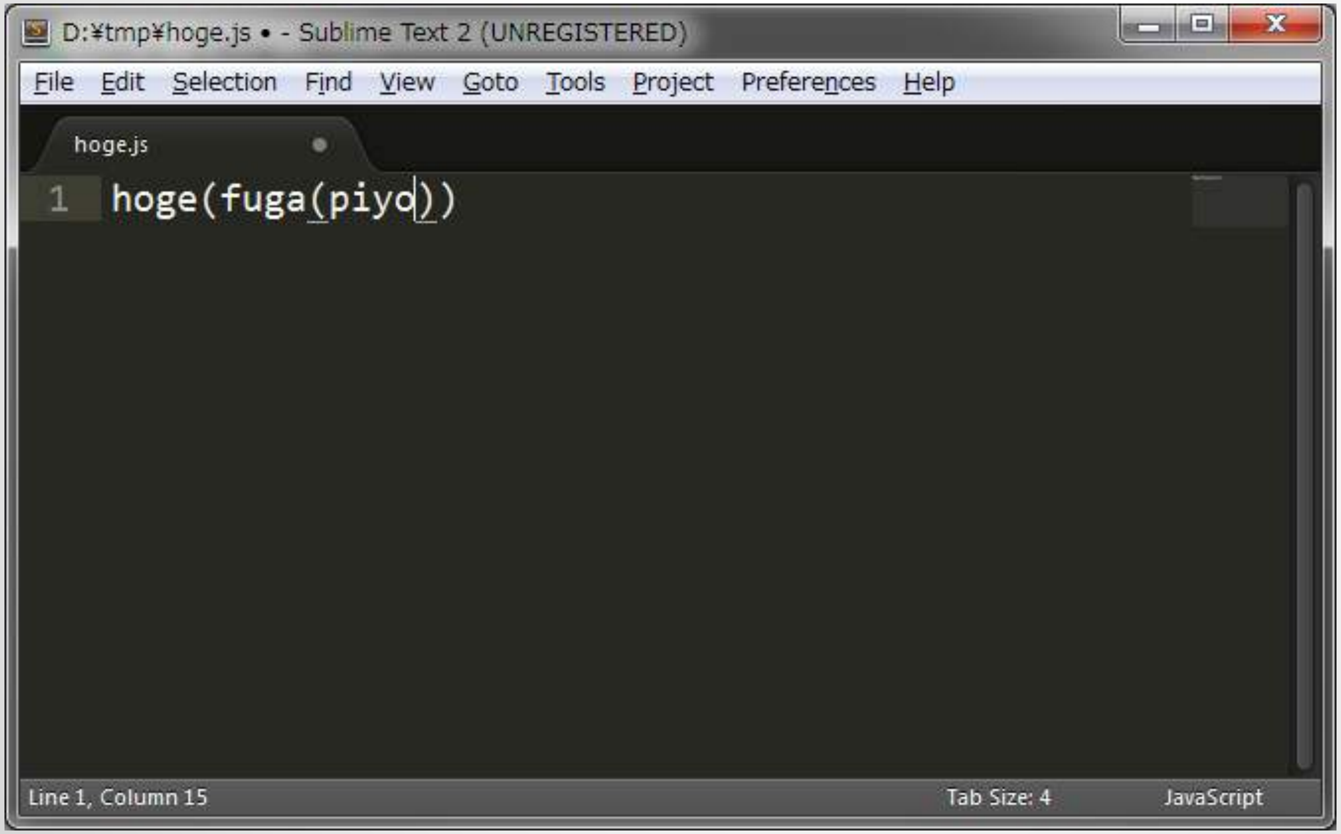
❖ Atom



```
foo(bar(buz))
```

For example

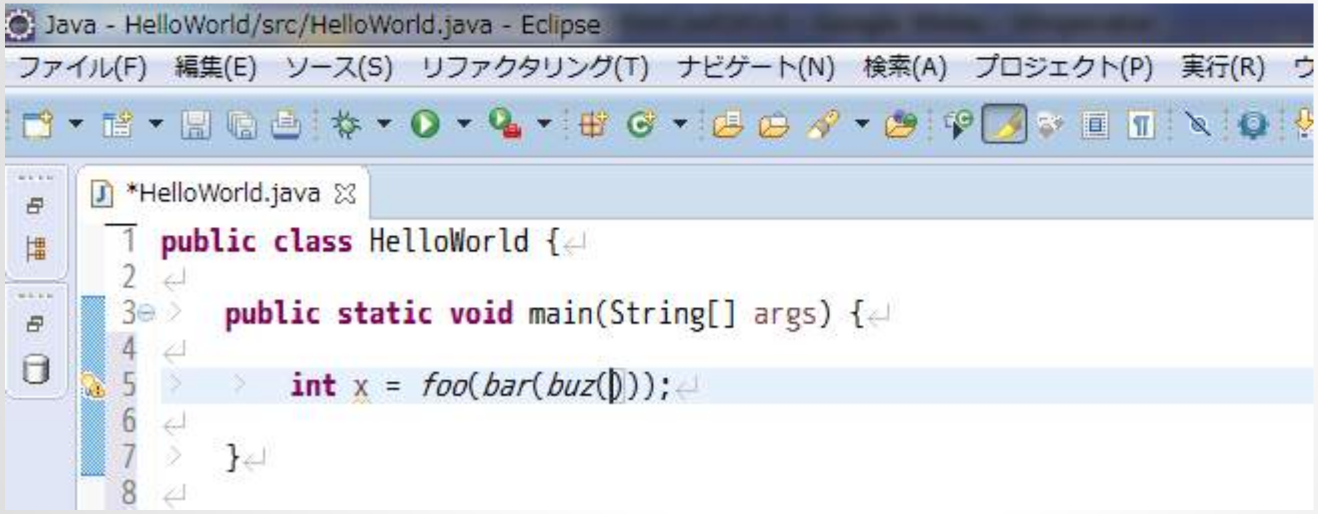
- Sublime Text


D:¥tmp¥hoge.js • - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
hoge.js
Line 1, Column 15
Tab Size: 4
JavaScript

```
hoge(fuga(piyo))
```

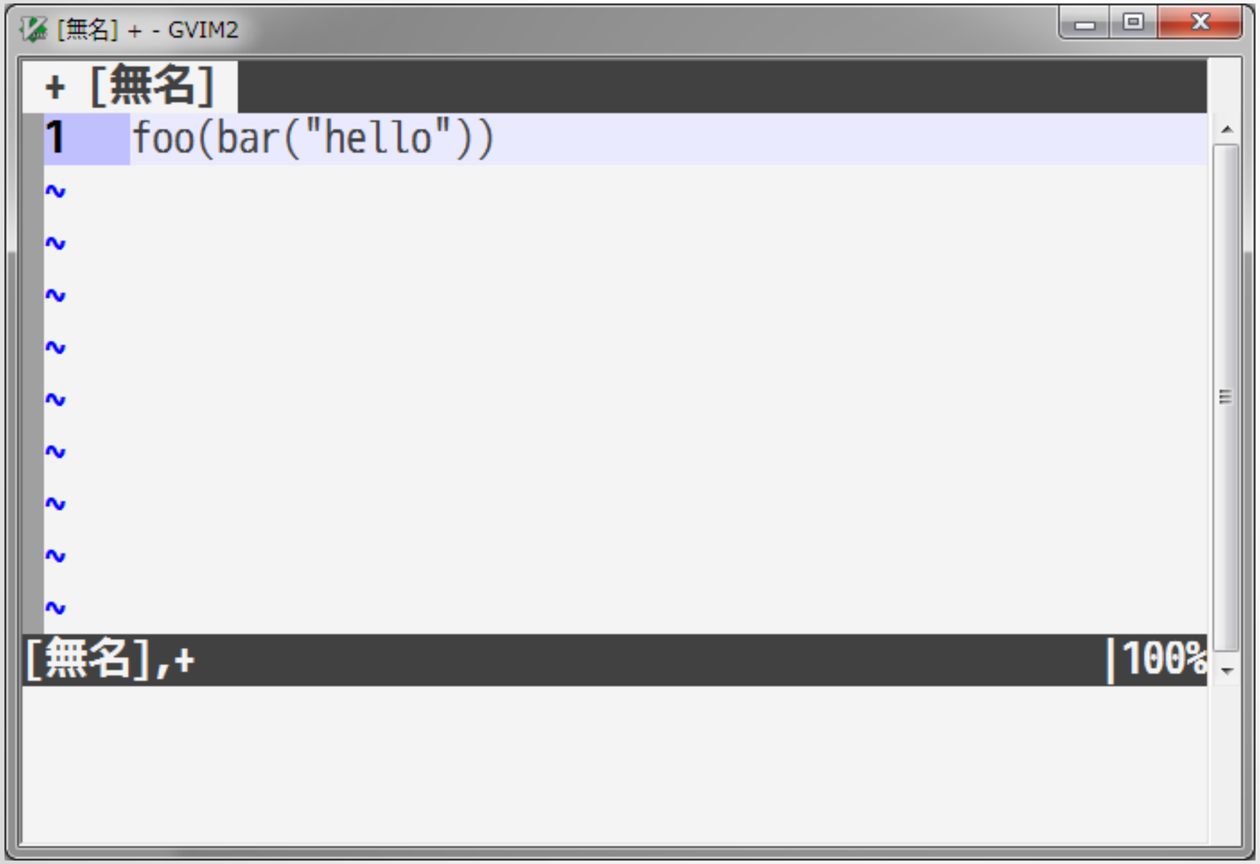
For example

❖ Eclipse


Java - HelloWorld/src/HelloWorld.java - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R)
*HelloWorld.java

```
public class HelloWorld {

    public static void main(String[] args) {

        int x = foo(bar(buz()));

    }

```

In Vim...



[無名] + - GVIM2

```
+ [無名]
foo(bar("hello"))
~
~
~
~
~
~
~
~
~
[無名],+                                                      |100%
```

Table of Contents

Table of Contents

Pure Vim way

```
inoremap  (  ()<Left>
inoremap  [  []<Left>
inoremap  "  ""<Left>
```

Very easy but ...

Pure Vim way but ...

```
let w = winnr())
```

```
let w = winnr)
```

```
let leftparen = "()"
```

```
I'm'
```

We need Plugin Power!

Auto-closing requirements

Before	Input	After
|	(	(|)
(|)	)	()|
(|)	<BS>	|
I|	'm	I'm|

syntax-specific rules

filetype-specific rules

Table of Contents

delimitMate

Raimondi/delimitMate

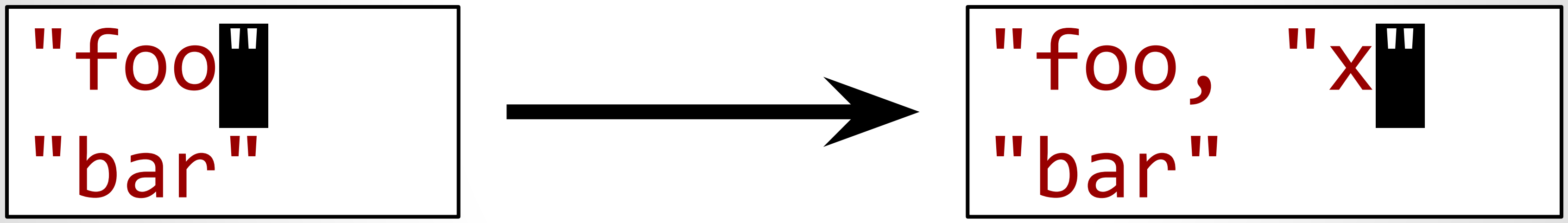
- some useful features

$$\begin{tikzcd} \boxed{\texttt{("foo|")}} \arrow[r, "\texttt{<C-G>g}"] & \boxed{\texttt{("foo")|}} \end{tikzcd}$$

- **unable to dot-repeat**

Unable to dot-repeat

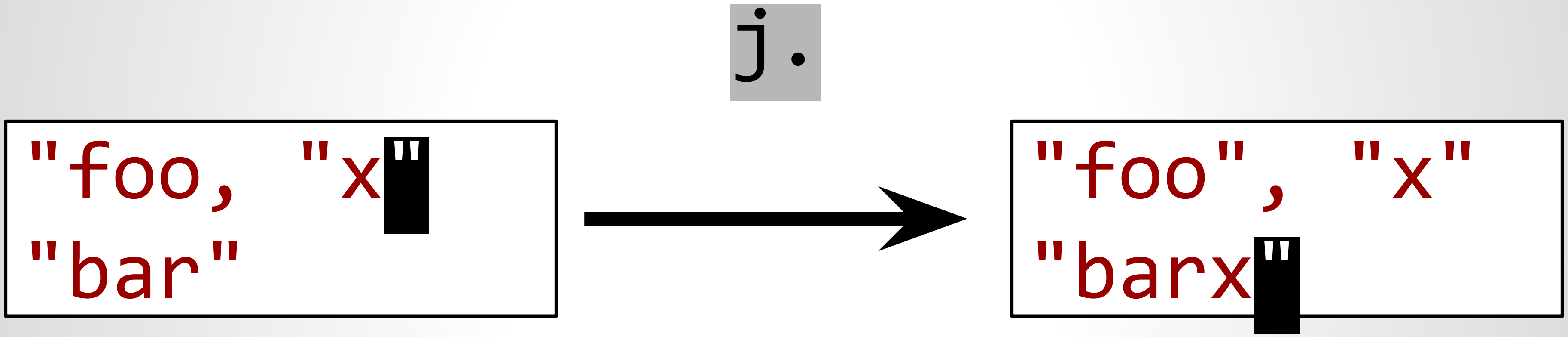
```
A, "x"<Esc>
```



```
"foo"
"bar"
```

```
"foo, "x"
"bar"
```

Unable to dot-repeat



```
j.
```

```
"foo, "x"
"bar"
```

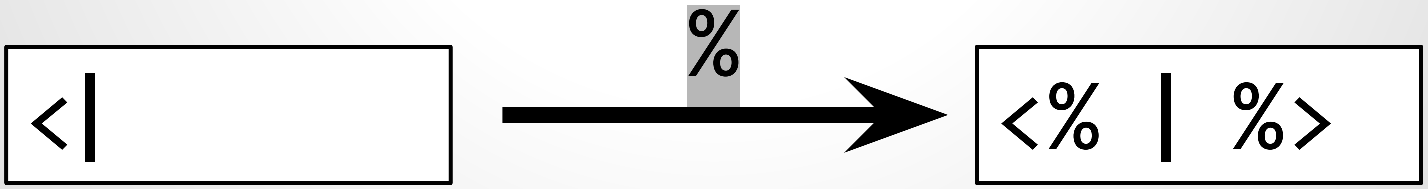
```
"foo", "x"
"barx"
```

vim-smartinput

kana/vim-smartinput

- highly customizable

```
call smartinput#define_rule({
\ 'at': '<\%#',
\ 'char': '%',
\ 'input': '%  %><Left><Left><Left>'})
```



- **unable to dot-repeat**

Table of Contents

About lexima.vim

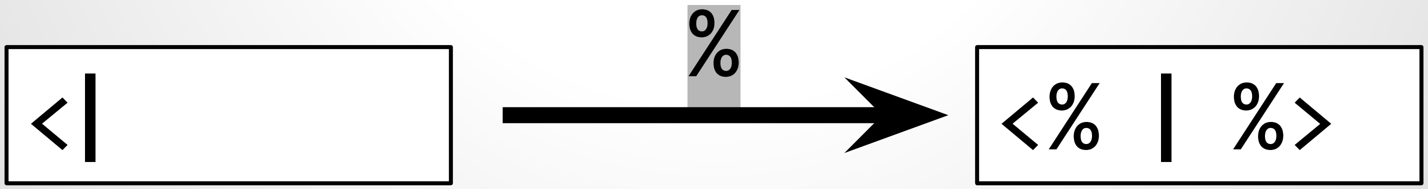
cohama/lexima.vim

- highly customizable (inspired by vim-smartinput)
- **dot-repeatable**
- unstable

lexima.vim's rule definition

inspired by kana/vim-smartinput

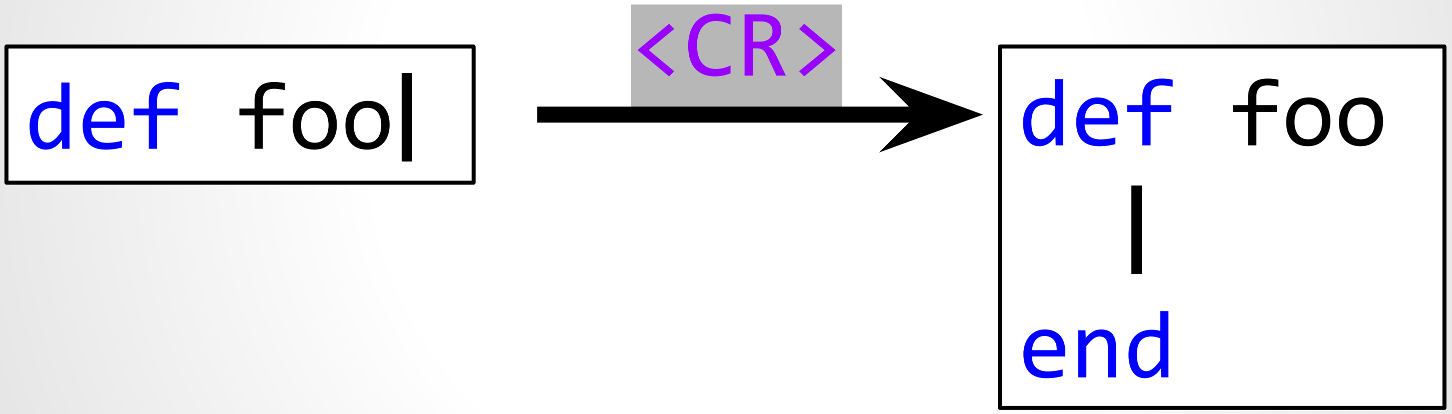
```vim
call lexima#add_rule({
\ 'at': '<\%#',
\ 'char': '%',
\ 'input': '% ',
\ 'input_after': ' %>'})
```



compatible with vim-smartinput

lexima.vim's endwise-rule

- inspired by tpope/vim-endwise



```
def foo|
```

```
def foo
  |
end
```

- Of course, dot-repeatable

Mechanism

In case of `(foo<Esc>`

`(|`	input `(`
`(|)`	call `setline()`
`(foo|)`	input `foo`
`(foo|`	call `setline()`
`(foo)|`	input `)`
`(foo|)`	call `setpos()`

Demo

Thanks