

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

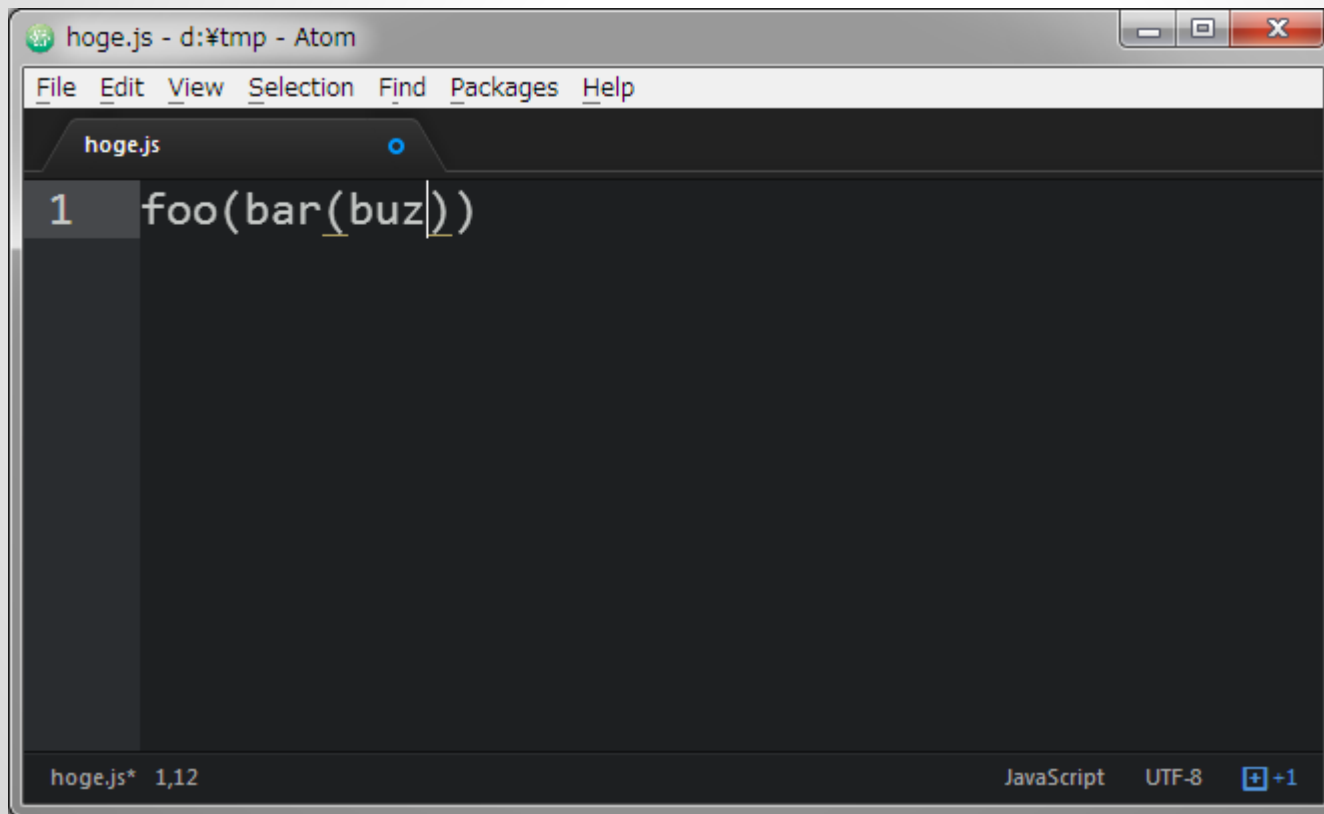
- ❖ We need auto-closing.

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

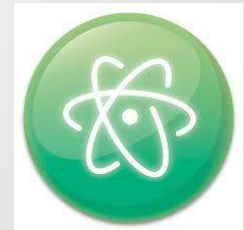
```
while (true) {  
    if (p == 1) {  
        foo();  
        break;  
    }  
}
```

For example

❖ Atom

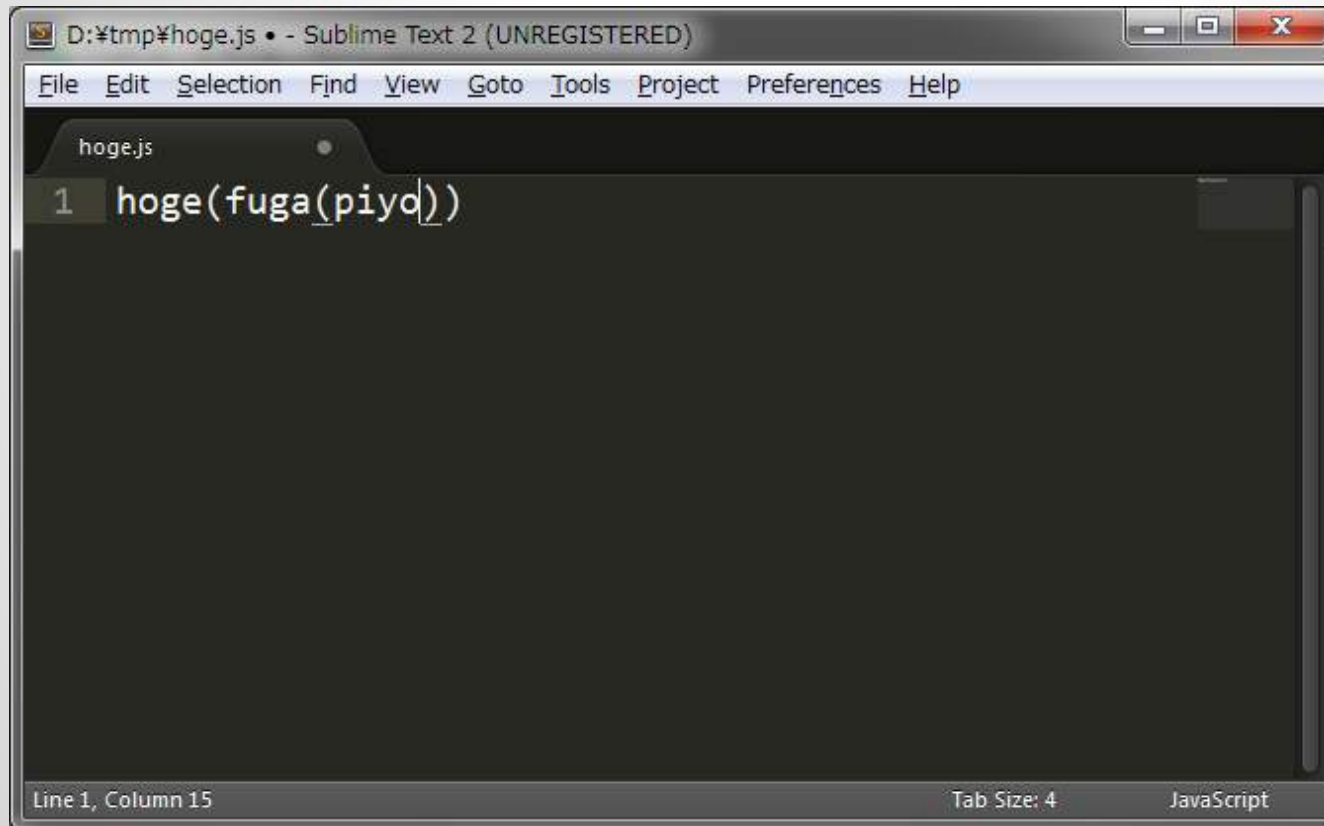


The image shows a screenshot of the Atom text editor. The window title is "hoge.js - d:\tmp - Atom". The menu bar includes "File", "Edit", "View", "Selection", "Find", "Packages", and "Help". The editor displays a single line of code: "1 foo(bar(buz))". The status bar at the bottom indicates "hoge.js* 1,12", "JavaScript", "UTF-8", and "+1".



For example

❖ Sublime Text



D:\tmp\hoge.js • - Sublime Text 2 (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

hoge.js

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

Line 1, Column 15 Tab Size: 4 JavaScript



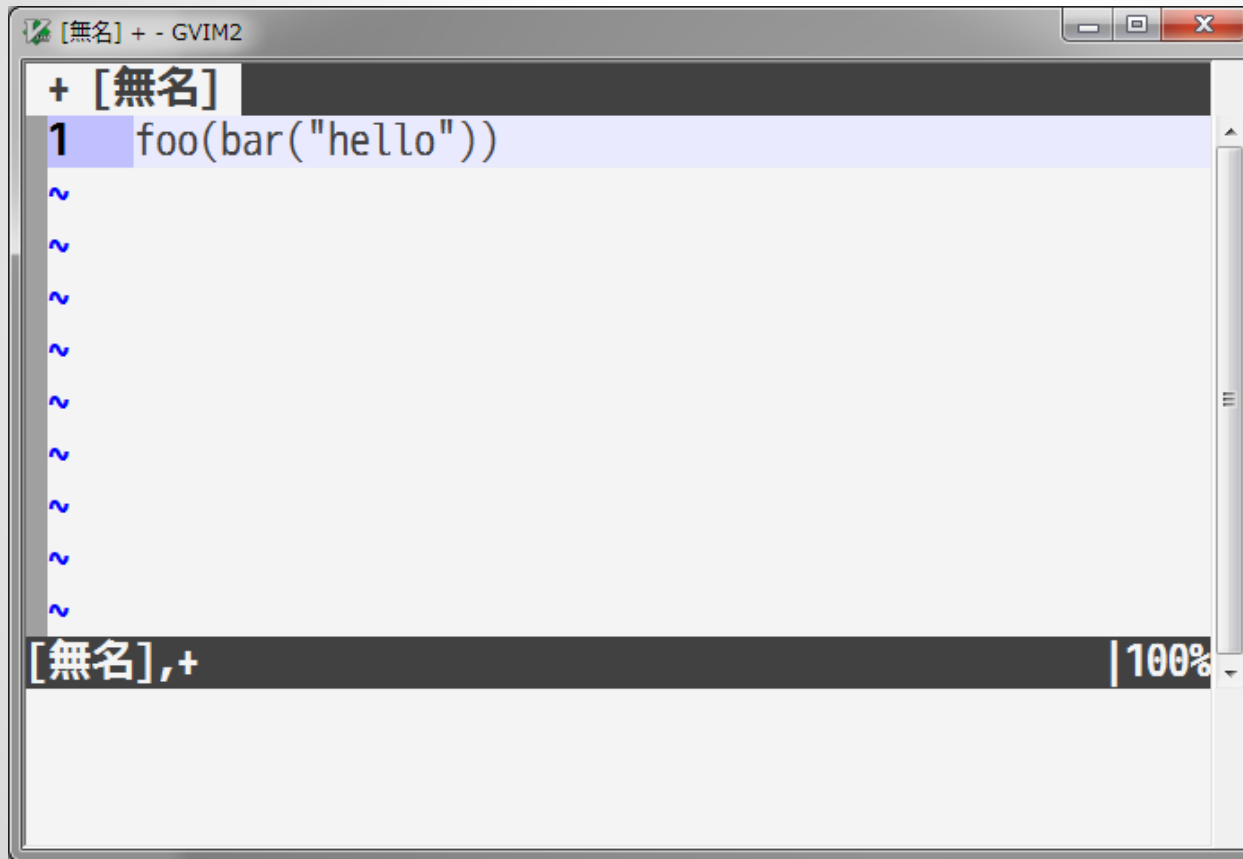
For example

❖ 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) ウ
*HelloWorld.java
1 public class HelloWorld {
2
3     public static void main(String[] args) {
4
5         > int x = foo(bar(buz(0)));
6
7     }
8
```


In Vim...



```
[無名] + - GVIM2
+ [無名]
1 foo(bar("hello"))
~
~
~
~
~
~
~
~
~
~
[無名],+ | 100%
```



Table of Contents

- ❖ Pure Vim way
- ❖ Some Plugins
- ❖ About `lexima.vim`

Table of Contents

- ❖ Pure Vim way
- ❖ Some Plugins
- ❖ About `lexima.vim`

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

- ❖ Pure Vim way
- ❖ Some Plugins
- ❖ About `lexima.vim`

delimitMate

Raimondi/delimitMate

- ❖ some useful features



- ❖ unable to dot-repeat

Unable to dot-repeat

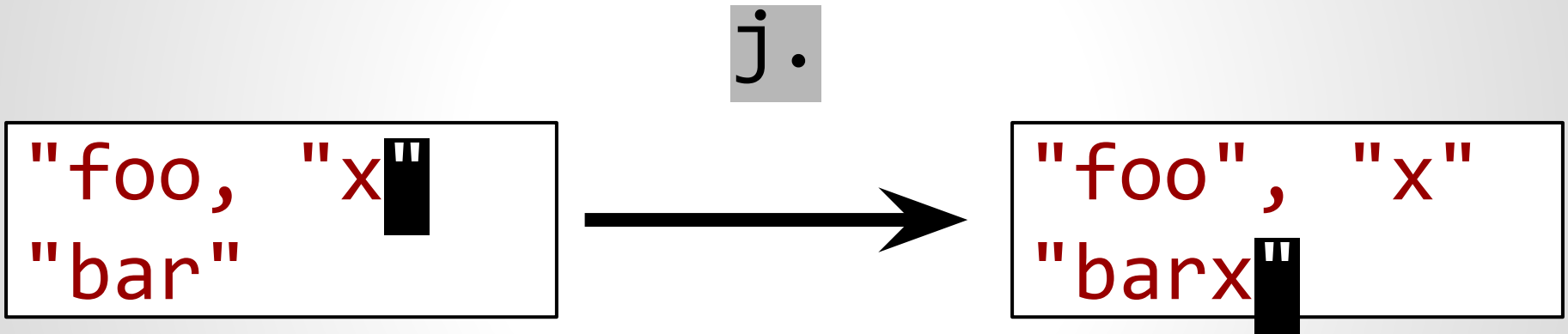
A, "x" <Esc>

"foo"
"bar"



"foo, "x"
"bar"

Unable to dot-repeat

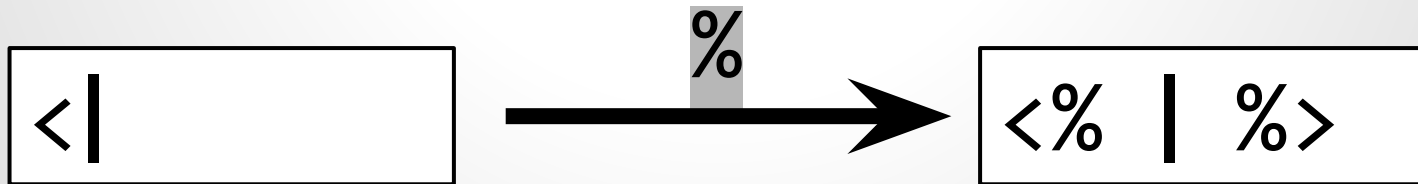


vim-smartinput

kana/vim-smartinput

- ❖ highly customizable

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



- ❖ unable to dot-repeat

Table of Contents

- ❖ Pure Vim way
- ❖ Some Plugins
- ❖ About `lexima.vim`

About `lexima.vim`

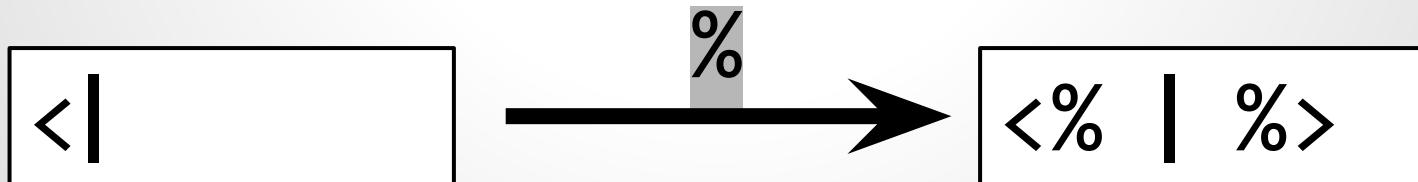
`cohama/lexima.vim`

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

lexima.vim's rule definition

inspired by kana/vim-smartinput

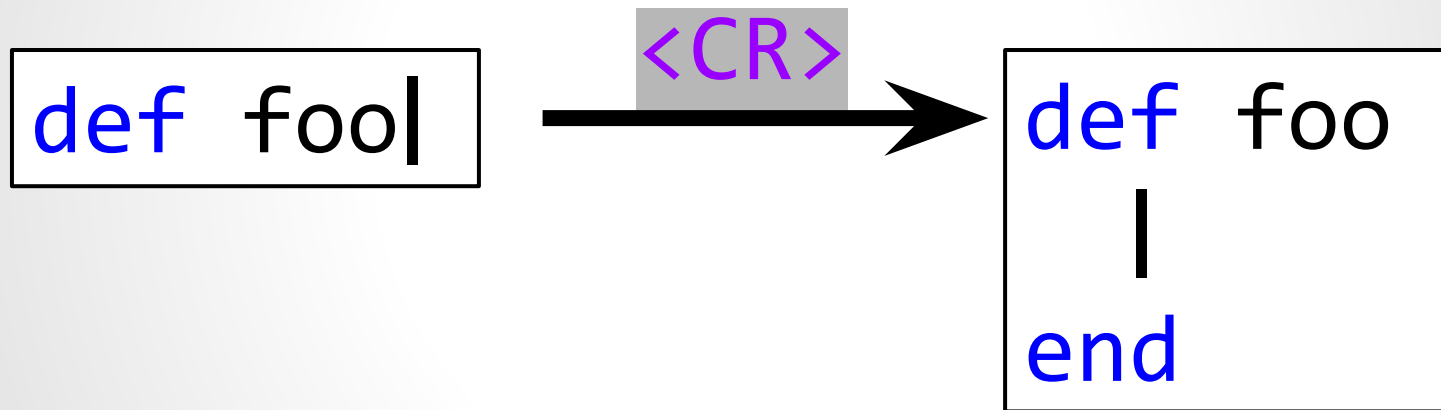
```
call lexima#add_rule({  
  \ 'at': '<\%#',  
  \ 'char': '%',  
  \ 'input': '% ',  
  \ 'input_after': ' %>'})
```



compatible with vim-smartinput

lexima.vim's endwise-rule

- ❖ inspired by tpope/vim-endwise



- ❖ Of course, dot-repeatable

Mechanism

In case of (foo<Esc>

(|

(|)

(foo|)

(foo|

(foo)|

(foo|)

input (

call **setline()**

input foo

call **setline()**

input)

call **setpos()**

Demo

Thanks