

# Vimscript

Sebastian Henneberg

**21. Juni 2010**

Die `.vimrc` enthält alle benutzerdefinierten Einstellungen

```
1 | set syntax
2 | set number
3 | set cul
4 | set autoindent
5 | set smartindent
6 | set encoding=utf-8
7 | filetype plugin on
8 | set textwidth=80
```

Die `.vimrc` enthält alle benutzerdefinierten Einstellungen

```
1 | set syntax
2 | set number
3 | set cul
4 | set autoindent
5 | set smartindent
6 | set encoding=utf-8
7 | filetype plugin on
8 | set textwidth=80

10 | " Um die "Danebenhauer" entschärfen ;)
11 | imap <F1> <ESC>
```

Die `.vimrc` enthält alle benutzerdefinierten Einstellungen

```
1 | set syntax
2 | set number
3 | set cul
4 | set autoindent
5 | set smartindent
6 | set encoding=utf-8
7 | filetype plugin on
8 | set textwidth=80

10 | " Um die "Danebenhauer" entschärfen ;)
11 | imap <F1> <ESC>
```

⇒ **Hierbei handelt es sich bereits um Vimscript!**

- ▶ imperatives Paradigma
- ▶ Typsystem
  - ▶ scalar
  - ▶ list
  - ▶ dictionary
- ▶ benutzerdefinierte Funktionen
- ▶ verschiedene Scopes
- ▶ Varargs
- ▶ Bedingungen
- ▶ Schleifen
- ▶ ternärer Operator
- ▶ call-by-value oder call-by-reference / in-place
- ▶ einfache Bindings

```
imap <C-D> <C-R>=strftime("%e %b %Y")<CR>
```

```
imap <C-T> <C-R>=strftime("%l:%M %p")<CR>
```

```
imap <C-D> <C-R>=strftime("%e %b %Y")<CR>
```

```
imap <C-T> <C-R>=strftime("%l:%M %p")<CR>
```

```
imap <C-C> <C-R>=string(eval(input("calc: ")))<CR>
```

```
imap <C-D> <C-R>=strftime("%e %b %Y")<CR>
```

```
imap <C-T> <C-R>=strftime("%l:%M %p")<CR>
```

```
imap <C-C> <C-R>=string(eval(input("calc: ")))<CR>
```

```
iabbrev ihc <C-R>='ich'<CR>
```

```
iabbrev werdne <C-R>='werden'<CR>
```

```
imap <C-D> <C-R>=strftime("%e %b %Y")<CR>
```

```
imap <C-T> <C-R>=strftime("%l:%M %p")<CR>
```

```
imap <C-C> <C-R>=string(eval(input("calc: ")))<CR>
```

```
iabbrev ihc <C-R>='ich'<CR>
```

```
iabbrev werdne <C-R>='werden'<CR>
```

```
iabbrev ... <C-R>='\cdots'<CR>
```

```
iabbrev OOP <C-R>=
```

```
\ 'objektorientierte Programmierung'<CR>
```

```
1 function! ToggleSyntax()  
2     if exists("g:syntax_on")  
3         syntax off  
4     else  
5         syntax enable  
6     endif  
7 endfunction
```

```
1 | function! ToggleSyntax()  
2 |     if exists("g:syntax_on")  
3 |         syntax off  
4 |     else  
5 |         syntax enable  
6 |     endif  
7 | endfunction  
  
2 | nmap ;s :call ToggleSyntax()<CR>
```

```
1 function! FileName()  
2     return expand('%:p')  
3 endfunction  
4  
5 function! CommentBlock(comment, author)  
6     let comment = "#"  
7     return repeat(comment,40) . "\<CR>"  
8     \ . comment . " " . FileName() . "\<CR>"  
9     \ . comment . "\<CR>"  
10    \ . comment . " " . a:author . "\<CR>"  
11    \ . comment . "\<CR>"  
12    \ . comment . " " . a:comment . "\<CR>"  
13    \ . repeat(comment,40) . "\<CR>"  
14 endfunction
```

- ▶ Einstellungen laden/ändern
- ▶ Syntax-Highlighting definieren
- ▶ Dateityp-Plugins laden
- ▶ automatische Vervollständigung erweitern
- ▶ automatische Einrückung erweitern
- ▶ Inhalt des Dokuments beliebig manipulieren
- ▶ eigene Befehle zusammen bauen
- ▶ auf viele vordefinierte Events reagieren
- ▶ Textschnipsel oder ganze Dokumente an Pipes senden
- ▶ externe Programme starten

- ▶ offizielle Dokumentation von Vimscript  
`http://vimdoc.sourceforge.net/html/doc/usr\_41.html`
- ▶ mehrteiliger Artikel über Vimscript  
`http://www.ibm.com/developerworks/linux/library/l-vim-script-1/index.html`
- ▶ Sammlung zahlreicher Vimscripts  
`http://www.vim.org/scripts/index.php`
- ▶ und natürlich...  
`:help vim-script-intro ;-)`