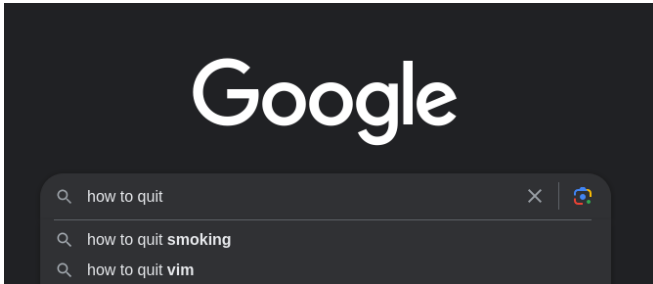


Vim

Darius Schefer

15. Juli 2023



Geschichtsstunde

- ▶ 1963: ed (Ken Thompson, Dennis Ritchie)
Ein Krampf
- ▶ 1976: ex und vi (Bill Joy)
Line Editor und Vollbild-Modus
- ▶ Danach: Elvis, nvi, Stevie ...
Diverse Klone
- ▶ 1991: vim (Bram Moolenaar)
Vi IMproved: Plugins, Unicode, Syntax Highlighting, ...
- ▶ 2015: Neovim (Internet)
LSP, Lua, async I/O, ...

Modales Editing

Verschiedene „Modi“

Anfangs verwirrend, wünscht man sich später überall

- ▶ Normal Mode
Standard Modus, Cursor und Text bewegen
- ▶ Insert Mode
Text eingeben
- ▶ Visual Mode
Text markieren
- ▶ Replace Mode
Text ersetzen
- ▶ Command Mode
Kommandos ausführen, Suchen

Motions im Normal Mode

h, j, k, l	links, runter, hoch, rechts
0, \$	Anfang und Ende der Zeile (und ^, g_)
w, e	Wortanfang oder -ende vorwärts
b, ge	Wortanfang oder -ende rückwärts
gg, G	Erste/letzte Zeile der Datei
%	Zur passenden Klammer springen
42G	Zeile 42
f, F	Vorwärts/rückwärts zu Buchstabe springen
t, T	Vorwärts/rückwärts vor Buchstabe springen
;, ,	Nächstes/vorheriges Match von f/F/t/T
/, ?	Vorwärts/rückwärts suchen
n, N	Nächstes/vorheriges Suchergebnis

Motions und Parameter

Die meisten Kommandos nehmen eine Zahl als Parameter!

- ▶ `5j` bewegt den Cursor 5 Zeilen nach unten
- ▶ `3w` bewegt den Cursor 3 Wörter vorwärts
- ▶ `2/wort` bewegt den Cursor zum 2. Vorkommen von „wort“ ab dem Cursor
- ▶ `2$` bewegt den Cursor zum Ende der nächsten Zeile
- ▶ und noch viele mehr

Operatoren im Normal Mode

Operatoren vor Motions führen eine Aktion auf der Region aus, über die der Cursor sich bewegt

d	Löschen (bzw. ausschneiden)
c	Löschen und neuen Text eingeben (Insert Mode)
y	Kopieren (p , P fügt oberhalb/unterhalb ein)
=	Einrücken
>	Nach rechts bewegen
<	Nach links bewegen

Jetzt wird's erst richtig lustig

Kombinationen aus Operatoren und Motions nehmen natürlich auch Parameter!

- ▶ `d3w` Löscht 3 Wörter „vorwärts“
- ▶ `y2/wort` Kopiert alles vom Cursor bis zum 2. Vorkommen von „wort“
- ▶ `dt{` Löscht alles bis zur nächsten „{“ (auf der aktuellen Zeile)

„Doppelte“ Operatoren arbeiten auf der Aktuellen Zeile:

`dd` löscht die aktuelle Zeile, `yy` kopiert sie.

Großbuchstabe heißt „bis Ende der Zeile“, `D` ist kürzer als `d$`

Text Objects

Vim kann Operationen „innerhalb von Objekten“ und „um Objekte herum“ ausführen

<code>iw</code>	Das aktuelle Wort
<code>aw</code>	Das aktuelle Wort und alles bis zum nächsten
<code>i(, i[, i{</code>	Innerhalb der aktuellen Klammern
<code>a(, a[, a{</code>	Inklusive der aktuellen Klammern
<code>ip, ap</code>	Aktueller Paragraph
<code>i", i'</code>	Innerhalb der aktuellen Anführungszeichen
<code>a", a'</code>	Inklusive der aktuellen Anführungszeichen

Insert Mode

Viele Commands bringen euch in den Insert Mode, mit `<ESC>` geht's zurück zum Normal Mode

<code>i</code>	Text links vom Cursor einfügen
<code>a</code>	Text rechts vom Cursor einfügen
<code>I</code>	Text vor erstem Zeichen der Zeile einfügen (<code>^i</code>)
<code>A</code>	Text nach letztem Zeichen der Zeile einfügen (<code>\$a</code>)
<code>s</code>	Zeichen unter dem Cursor ersetzen (<code>xi</code>)
<code>o, O</code>	Neue Zeile unter/über der aktuellen

`3iabc` fügt „abcabcabc“ ein nachdem ihr in den Normal Mode zurückkehrt

Visual Mode

Diese Tasten beginnen eine Text-Auswahl, auch die lässt sich mit `<ESC>` abbrechen

<code>v</code>	Auswahl beginnen
<code>V</code>	„Visual-line“ Mode
<code><C-v></code>	„Visual-block“ Mode
<code>o</code>	Zum anderen Ende der Auswahl springen

Mit `gv` wählt ihr eure letzte Textauswahl nochmal aus

Text ersetzen

`ra` im Normal Mode ersetzt das Zeichen unter dem Cursor mit „a“

`ra` im Visual Mode die gesamte Auswahl!

Replace Mode ist keine extra Folie wert, drückt ihr im Normal Mode `R`, überschreibt ihr euren Text mit neuen Zeichen bis ihr wieder `<ESC>` drückt.

Oh no

Im gegensatz zu vi gibt es im Jahr 1991 multi-level undo

u	Letzte Bearbeitung rückgängig machen
<C-r>	Das Rückgängigmachen rückgängig machen
U	!?!?

U laut Anleitung: „Undo all latest changes on one line, the line where the latest change was made. U itself also counts as a change, and thus U undoes a previous U.“

Einfach nicht drücken, und wieder vergessen.

Commands und Command Mode

: im Normal Mode bringt euch in den Command Mode und wartet auf einen Befehl:

:w	Speichert die aktuelle Datei
:w a.txt	...unter dem Namen „a.txt“
:q	Schließt den Editor
:q!	...bei ungespeicherten Veränderungen (ZQ)
:wq, :x	Speichern und Schließen (ZZ)
:r a.txt	Kopiert den Inhalt von a.txt hierher
:set <option>	Setzt eine Einstellung

Habt ihr bereits Text im Visual Mode ausgewählt und drückt dann :, wird dies direkt zu :’<, ’> erweitert.

Der Befehl (z.B. :’<, ’>w) speichert dann nur eure Auswahl in einer Datei.

Suchen und ersetzen

Der `:s` Befehl kann Text ersetzen

<code>:s/a/b/</code>	Ersetzt das erste „a“ der Zeile mit „b“
<code>:s/a/b/g</code>	Ersetzt alle „a“, lies g als „global“
<code>:%s/a/b/</code>	Führt <code>:s/a/b/</code> auf jeder Zeile aus
<code>:%s/a/b/g</code>	Ersetzt alle „a“ in der Datei durch „b“
<code>:%s/a/b/gc</code>	Fragt für alle „a“ nach Bestätigung („confirm“)

`:s` kann sogar reguläre Ausdrücke!

RTFM

Vim kommt mit einer eingebauten Anleitung!

<code>:h</code>	Öffnet die Anleitung
<code>:h w</code>	Sagt euch was <code>w</code> macht
<code>:h 'number'</code>	Sagt euch was <code>:set number</code> macht
<code>:h :h</code>	Zeigt euch die Anleitung zur Anleitung

Die Anleitung zu lesen lohnt sich, vim kann so viel mehr als in einen Vortrag passt!

Falls noch Zeit ist

- ▶ Die `$EDITOR` und `$VISUAL` Umgebungsvariablen
- ▶ Neovim als `$PAGED` und `$MANPAGER`
- ▶ Die `.vimrc`
- ▶ Register und Makros
- ▶ Completion

Hoffentlich ist noch Zeit für

Fragen?

:wq

Danke!