

## VIM – ANFÄNGER




Grau: andere Variante des Befehls darüber  
Klein: Hinweise / Merkhilfen

Legende:

{dn} Dateiname/Dateipfad

{z} Ein Zeichen



auch in Visual Studio Code (oben ) , Visual Studio 2022 CE (mitte ) , Eclipse (unten )

{n} Eine Zahl

{d} Eine Dezimalziffer

{t} Text







{b} Bewegungsbefehl

{r} Register-Zeichen (u.a. a-z/A-Z)

Alternative Möglichkeit

Varianten des Befehls







### Moduswechsel, Speichern, Beenden

→NORMAL - Aus jedem anderen Modus	Esc	
→BEFEHL (z.B. :w /such ?rückwärts)	: / ?	
Datei speichern (mit vorhandenen Namen) Write	:w	
Datei speichern (Name festlegen/neuer Name)	:saveas {dn}	
Dokument schließen & Schließen erzwingen Quit	:q & :q!	
Speichern und schließen	:wq	

### Text eingeben (Einfügen, Ersetzen)

→EINFÜGEN - auf dem Cursor Insert	i	
→EINFÜGEN - rechts vom Cursor Append	a	
→EINFÜGEN - Ende der Zeile	A	
→EINFÜGEN - Anfang der Zeile	I	
→EINFÜGEN - Neue Zeile unterhalb	o	
→EINFÜGEN - Neue Zeile oberhalb	O	
→ERSETZEN - auf dem Cursor Replace	R	
Zeichen unter Cursor ersetzen	r{z}	

### Text markieren / Kopieren / Ausschneiden









→VISUELL - Text markieren	v	{v{b}	
→VISUELL - Zeilen markieren	V	{V{b}	
Markierte(r) Zeilen/Text kopieren	y		
Aktuelles Wort markieren	yiw		
Details zu iw bei Löschen			
Markierte(r) Zeilen/Text ausschneiden Delete	d		
→EINFÜGEN Markierte(r) Zeilen/Text löschen Change	c		

### Rückgängig / Wiederherstellen (Undo/Redo)








Rückgängig (Undo)	u	:undo	
Rückgängig (komplette Zeile)	U		
Wiederherstellen (Redo)	Strg+R	:redo	

↪EINFÜGEN	Wechselt in den Einfüge-Modus (INSERT), dort wird Text eingefügt
↪ERSETZEN	Ersetzen-Modus (REPLACE), dort wird Text ersetzt
→VISUELL	Visuell-Modus (VISUELL), dort wird Text markiert
→NORMAL	Normal-Modus, hier funktionieren Tastenbefehle (z.B. \$)
→BEFEHL	Befehls-Modus, hier können längere Befehle eingegeben werden Befehle in diesem Modus werden inkl. dem :, / oder ? angegeben.

### Cursor Bewegen (diese Befehle sind {b}-Bewegungsbefehle)

Cursor bewegen (links runter hoch rechts)	h j k l	
Merken mit: h ist links, j sieht aus wie ein Pfeil nach unten		
Cursor Wortweise bewegen (zum Wortanfang)	w	
Word		
Cursor Wortweise bewegen (zum Wortende)	e	
End		
Cursor Wortweise bewegen (ein Wort zurück)	b	
Back		
Cursor zum Zeilenanfang	0	
Cursor zum Zeilenende	\$	
Cursor zum Zeilenanfang, erstes Zeichen	^	
Einfach ^-Taste zweimal drücken		
Cursor zum Dokumentanfang	gg	
Cursor zum Dokumentende Shift+G, Shift ist unten = Ende	G	
Cursor zur Zeile {n}	{n}G	
Bewegt den Cursor nach Oben Mitte Unten	H M L	
Vom Fenster, Text bleibt unverändert High Middle Low		

### Text/Zeichen suchen (diese Befehle sind {b}-Bewegungsbefehle; bis auf :noh)

Zeichensuche Vor- & Rückwärts, auf Zeichen Find	f{z} & F{z}	
Zeichensuche Vor- & Rückwärts, vor Zeichen To	t{z} & T{z}	
Zeichensuche Wiederholen & Zurück	; & ,	
Textsuche Vorwärts & Rückwärts	/ {t} & ? {t}	
Textsuche Wiederholen & Zurück Next	n & N	
Cursor zur passenden Klammer (⇔), [⇔], {⇔}, <⇔>	%	
Markierung der gefundenen Texte entfernen	:noh	
No Highlight		

VIM – ANFÄNGER

Löschen

Beim löschen mit d... wird der gelöschte Text ausgeschnitten, d.h. er kann mit p/P eingefügt werden.  
iw (Inner Word) funktioniert bei allen Befehlen, die ein Bewegungsbefehl erwarten. iw ohne einem Vorangestellten Befehl (diw, viw, yiw) würde beim i in den Einfügemodus wechseln und ein w einfügen!

Zeichen unter Cursor } vor dem Cursor löschen	x	} X
Aktuelle Zeile löschen	dd	
Löschen bis {b}-Bewegungsbefehl z.B. d%, 5dj (löscht 5 Zeilen)	Delete	d{b}
Bis Zeilenende löschen	D	
Aktuelles Wort löschen (Cursorposition egal) Kombination aus d (Delete), i (Inner) und w (Wortende)		diw
Text ändern (wechselt in Einfüge-Modus)	Change	c } c{b}

Kopieren

Aktuelle Zeile kopieren	copY	Yy
Kopieren bis {b}-Bewegungsbefehl		y{b}
Kopieren in Register {r} Funktioniert auch mit {b} : "{r}y{b} Register a-z/A-Z können frei belegt werden. d,y kopiert ohne Angabe nach "-Register (entspricht ""y). + ist die Zwischenablage (""+y)		"{r}y
Aktuelles Wort kopieren (Cursorposition egal)		yiw

Einfügen

Einfügen, nach Cursorposition	p
Einfügen, über Cursorposition	P
Einfügen, aus Register	"{r}p ⤵ "{r}P
Einfügen aus Zwischenablage	"+p ⤵ "+P

Konfiguration

vim ~/.vimrc	öffnet/erstellt die Konfigurationsdatei	↓ dies hier eintragen ↓
Zeilennummer und Relative Zeilennummer	set nu set rnu	
Am Ende/Anfang der Zeile zur Zeile davor/danach	set wrap set backspace=indent,eol,start set whichwrap=<,>,h,l[,]	
Tabstop/Einrücken Abstand definieren	set tabstop=4 set shiftwidth=4	
Einrücken aktivieren (und Space statt Tab)	set autoindent set expandtab	

Register

Zuletzt (0) und davor (1...9) kopierte Texte	0 ... 9
Register ohne Namen (für y, d)	"
Dateinamen	% #
Zwischenablage	+
Letztes Suchmuster	/
Letzte Kommandozeile	:
Zuletzt eingefügter/geschriebener Text	.
Zuletzt gelöscht (weniger als eine Zeile!)	-
Ausdruck-Register	=
Schwarzes Loch Register	_
Frei wählbare Register	a...Z

mehrfaches Ausführen von Befehlen

Vor jedem Befehl im Normal-Modus kann eine beliebig lange Zahl vorangestellt werden. Diese besagt, wie oft der Befehl (nach dessen Beendigung) ausgeführt werden soll.

So fügt 100i -<ESC> 100 mal das - Zeichen ein. 20j geht 20 Zeilen nach unten.  
Ein sehr mächtiges Werkzeug.

Bewegungsbefehle richtig nutzen

Um einen Text zu kopieren, muss dieser nicht zwangsweise markiert werden. Es reicht y zu drücken und dann ein Bewegungsbefehl, der das Ziel eindeutig bestimmt. Beispiel:

```
list.Add(nextUserName);  
  
fnct)previousUserName<ESC>  
Springt zum n, entfernt alles bis vor der ), schreib previousUserName und wechselt zurück in den Normal-Modus → list.Add(previousUserName);
```





Relative Zeilennummer

Mit set rnu werden Relative Zeilennummer angezeigt. Diese können dann schnell mit z.B. 15j (15 Zeilen runter) der 15k (15 Zeilen hoch) angesprungen werden.

	↓ dies hier eintragen ↓
Aktuelle Zeile hervorheben	set cursorline
Sonstiges (Position anzeigen, Highlight-Search)	set ruler set hlsearch

## VIM – FORTGESCHRITTEN








Grau: andere Variante des Befehls darüber  
Klein: Hinweise / Merkhilfen

Legende:  auch in Visual Studio Code (oben ) , Visual Studio 2022 CE (mitte ) , Eclipse (unten ) {dn}  
Dateiname/Dateipfad {n} Eine Zahl {d} Eine Dezimalziffer ↵ Alternative Möglichkeit  
{z} Ein Zeichen {t} Text {b} Bewegungsbefehl {r} Register-Zeichen (u.a. a-z/A-Z) ⏏ Variante des Befehls







### Befehle im Eingabemodus

Entfernt Wort unter Cursor	Strg+w	
Einrücken ⏏ Zurückrücken	Strg+t ⏏ Strg+d	
Inhalt eines Registers einfügen	Strg+r{n}	

### Text eingeben (Einfügen, Ersetzen)

➔EINFÜGEN - Ende der Zeile	A	
➔EINFÜGEN - Anfang der Zeile	I	
➔EINFÜGEN - Zeichen unter Cursor löschen	s	
➔EINFÜGEN - aktuelle Zeile löschen	S	
Substitute ↵ CC		
➔EINFÜGEN - Ersetzen bis {b}-Bewegungsbefehl	c{b}	
Change		
➔EINFÜGEN - Zeile ab Cursor löschen	C	
➔EINFÜGEN- Aktuelle Wort ersetzen (Cursorposition egal)	ciw	

### Text markieren / Text in Markierung verändern

↵VISUELL - Markieren ⏏ Zeilenweise markieren	v ⏏ V	
↵VISUELL - Blockweise markieren	Strg+V	
Im Visuell-Modus Wort unter Cursor markieren (egal wo der Cursor im Wort steht)	aw	
Text Einrücken ⏏ Zurückrücken	> ⏏ <	
Markierten Text Groß-/Kleinschreibung umschalten	~	
Markierten Text in Groß- ⏏ Kleinbuchstaben	u ⏏ U	

### Rückgängig / Wiederherstellen (Undo/Redo)

Rückgängig (Undo)	u ↵ :undo	
Rückgängig (komplette Zeile)	U	
Wiederherstellen (Redo)	Strg+R ↵ :redo	

Wechselt in den  
➔EINFÜGEN Einfüge-Modus (INSERT), dort wird Text eingefügt  
↵ERSETZEN Ersetzen-Modus (REPLACE), dort wird Text ersetzt  
↵VISUELL Visuell-Modus (VISUELL), dort wird Text markiert  
➔NORMAL Normal-Modus, hier funktionieren Tastenbefehle (z.B. \$)  
➔BEFEHL Befehls-Modus, hier können längere Befehle eingegeben werden  
Befehle in diesem Modus werden inkl. dem :, / oder ? angegeben.

### Cursor Bewegen (diese Befehle sind {b}-Bewegungsbefehle)

Cursor zum Zeilenanfang	0	
Cursor zum Zeilenende	\$	
Cursor zum Zeilenanfang, erstes Zeichen	^	
Einfach ^-Taste zweimal drücken		
Cursor zum Zeilenende, letztes Zeichen	g_	
Bewegt den Cursor nach Oben Mitte Unten	H M L	
Vom Fenster, Text bleibt unverändert High Middle Low		
Cursor 1 Wort zurück, zum Wortende gE das gleiche aber mit Interpunktion.	ge	
Cursor an Block {...} Anfang ⏏ Ende [{ ⏏ }] erneut geht eine Blockebene runter	[{ ⏏ }]	
Nächster Absatz ⏏ Vorheriger Absatz	} ⏏ {	

### Text suchen / Ersetzen (diese Befehle sind {b}-Bewegungsbefehle; bis auf :noh)

Wort unter Cursor suchen vorwärts ⏏ rückwärts	* ⏏ #	
Suchen/Ersetzen - Zeilenweise, einmal (t) Was wird gesucht/(t) Ersetzungstext/{n} Bei nächsten n-Zeilen Ohne {n} nur bei der aktuellen Zeile.	:s/{t}/{t}/{n}	
Suchen/Ersetzen - Zeilenweise, komplette Zeile g steht für Global, also für alle Vorkommen	:s/{t}/{t}/g[{n}]	
Suchen/Ersetzen - komplette Dokument	:%s/{t}/{t}/g	
Suchen/Ersetzen - Nachfragen vor dem Ersetzen Das c steht für Check	:%s/{t}/{t}/gc	
Markierung der gefundenen Texte entfernen No Highlight	:noh	

## VIM – FORTGESCHRITTEN

### Scrollen

Eine Zeile hoch ↵ runter scrollen	Strg+E ↵ Strg+Y	
Eine halbe Seite hoch ↵ runter scrollen	Strg+D ↵ Strg+U	down/up
Eine ganze Seite hoch ↵ runter scrollen	Strg+F ↵ Strg+B	
Text scrollen das Cursor oben ist ↵ Erstes Zeichen	zt ↵ z<Return>	
Text scrollen das Cursor zentriert ist ↵ erst. Zeichen	zz ↵ z.	
Text scrollen das Cursor unten ist ↵ Erstes Zeichen	zb ↵ z-	
Text scrollen das Zeile {n} oben/zent./unten ist	{n}zt ↵ {n}zz ↵ {n}zb	

<b>Löschen / Kopieren / Einfügen</b> Löschen bis {b} ↵ ganze Zeile	d{b} ↵ dd	
In Register	"{r}d{b} ↵ "{r}dd	
Kopieren bis {b} ↵ ganze Zeile	y{b} ↵ yy	
In Register	"{r}y{b} ↵ "{r}yy	
Einfügen vor ↵ nach Cursor / Zeile	p ↵ P	
Aus Register	"{r}p ↵ "{r}P	

### Register

Alle Register anzeigen	:reg	
Bestimmtes Register anzeigen	:reg {z}	
Zuletzt (0) und davor (1...9) kopierte Texte	0 ... 9	
Register ohne Namen (für y, d)	"	
Dateinamen	% #	
Zwischenablage	+	
Letztes Suchmuster	/	
Letzte Kommandozeile	:	
Zuletzt eingefügter/geschriebener Text	.	
Zuletzt gelöscht (weniger als eine Zeile!)	-	
Ausdruck-Register	=	
Schwarzes Loch Register	_	
Frei wählbare Register	a...z A...Z	

### Ausrichten (automatisch einrücken)

Aktuelle Zeile ↵ Zeile bis {b}-Bewegung ausrichten	== ↵ ={b}	
Komplettes Dokument ausrichten	gg=G	

### Einrücken

Einrücken ↵ Zurückrücken	>> ↵ <<	
10>> rückt nicht 10 mal ein, sondern 10 Zeilen werden eingerückt.		
Einrücken bis passende Klammer	>% ↵ <%	
(Cursor steht auf einer Klammer (), {}, [], <>)		
Einrücken bis {b}-Bewegung	>{b} ↵ <{b}	
Bewegungen können auch inner/outer s.u. sein!!!		

### Inner / Outer (diese Befehle sind {b}-Bewegungsbefehle)

Einer der mächtigsten Bereichsangaben überhaupt!

Inner, nur der innere Teil des Bereichs	{Befehl}i{Typ}	
Outer, inkl. das umschließende Element	{Befehl}a{Typ}	
Befehle: kopieren ↵ löschen ↵ ersetzen ↵ markieren	y ↵ d ↵ c ↵ v	
Typ - Block (...)	vib ↵ vi( ↵ vi)	
Die Beispiele mache ich nur für markieren (v) und nur mit inner (i). Bei den anderen Befehlen funktioniert es genauso. Auch in Verbindung mit Register "ayvi( → Kopiert den Text innerhalb des Blocks in Register a. Mit den outer (o) funktioniert es dann ebenfalls gleich.		
Typ - Block {...}	viB ↵ vi{ ↵ vi}	
Typ - Wort	viw	

### Texte verändern

Aktuelle Zeile klonen (nach unten)	Strg+l ↵ yyp	
Die Zeile darunter zur aktuellen hinzufügen	J ↵ gJ	
Groß- und Kleinbuchstaben umschalten	g~{b}	
In Groß- ↵ Kleinbuchstaben ändern	gU{b} ↵ gu{b}	

### Inner/Outer - Erklärungen und Beispiele





Die Inner und Outer sind nützliche Möglichkeiten komplette Bereich zu markieren, kopieren, löschen, egal wo innerhalb des Bereichs der Cursor steht.

Aktuellen Codeblock markieren ↵ kopieren ↵ löschen	vi{ ↵ yi{ ↵ di{
Für Registerzuweisung einfach "{r} voranstellen	"ayi{ ↵ "adi{
Auch mit Anzahl: Markiere 5 Wörter	5vaw
Hier ist outer wichtig, da sonst jedes Leerzeichen als ein Wort zählt.	
Klammer, aktuelle oder nächste	vi(
Bei Klammern (...) kann die aktuelle (Cursor befindet sich darin) oder die vom Cursor aus Nächste genutzt werden.	
	Cursor steht innerhalb einer Klammer oder irgenwo davor.

## VIM – PROFIS

Grau: andere Variante des Befehls darüber  
Klein: Hinweise / Merkhilfen





Legende:

 auch in Visual Studio Code (oben ) , Visual Studio 2022 CE (mitte ) , Eclipse (unten )  
{dn} Dateiname/Dateipfad {n} Eine Zahl {d} Eine Dezimalziffer ↵ Alternative Möglichkeit  
{z} Ein Zeichen {t} Text {b} Bewegungsbefehl {r} Register-Zeichen (u.a. a-z/A-Z) ⚡ Variante des Befehls





### Allgemein

Hilfe öffnen	:help {t}	
Ansicht schließen	:close	
Terminal (PowerShell, Cmd, Bash) öffnen	:terminal	
Alle Dateien schließen (erzwingen)	:qa!	
Alle Dateien speichern ⚡ und beenden	:wa ⚡ :wqa	








### Befehle im Eingabemodus

Entfernt ein Zeichen <b>nach</b> dem Cursor (Backspace)	Strg+h	
Neue Zeile (Return)	Strg+j	
Nächstes ⚡ vorheriges passendes Wort einfügen Autovervollständigung / Intellisense	Strg+n ⚡ Strg+p	
Für einen Befehl in Befehlsmodus wechseln z.B. Strg+o ciw	Strg+o	

### Text markieren / Text in Markierung verändern





Im Visuell-Modus Cursor an Anfang/Ende der Mark.	O	
Im Visuell-Modus Blöcke (...) markieren	a( ⚡ i( ↵ ab ⚡ ib	
Im Visuell-Modus Blöcke {...} markieren	a{ ⚡ i{ ↵ aB ⚡ iB	
Im Visuell-Modus Blöcke <...> markieren	a< ⚡ i< ↵ at ⚡ it	

### Vergleichen










Vim im Vergleichsmodus öffnen	vim -d {file1} {file2}	
Zur nächsten ⚡ vorherigen Abweichung springen	]c ⚡ [c	
Die Differenz des anderen Puffer zum aktuellen kopieren (get)	do ↵ :diffget	
Die Differenz des aktuellen Puffers zum anderen senden (put)	.Ddp ↵ :diffput	
Aktuelles Fenster zum Vergleichsfenster machen Dann Fenster teilen/Datei laden/:diffthis	:diffthis	
Vergleich aktualisieren	:difffupdate	
Vergleichsmodus beenden	:diffoff	

↵ EINFÜGEN	Wechselt in den Einfüge-Modus (INSERT), dort wird Text eingefügt
↵ ERSETZEN	Ersetzen-Modus (REPLACE), dort wird Text ersetzt
~ VISUELL	Visuell-Modus (VISUELL), dort wird Text markiert
→ NORMAL	Normal-Modus, hier funktionieren Tastenbefehle (z.B. \$)
→ BEFEHL	Befehls-Modus, hier können längere Befehle eingegeben werden Befehle in diesem Modus werden inkl. dem :, / oder ? angegeben.

### Cursor Bewegen (diese Befehle sind {b}-Bewegungsbefehle)

Bewegt den Cursor nach Oben Mitte Unten	H M L	
Cursor an Block {...} Anfang ⚡ Ende [{ ⚡ }] erneut geht eine Blockebene runter	[{ ⚡ }]	
Nächster Absatz ⚡ Vorheriger Absatz	} ⚡ {	
Zur Deklaration ⚡ globale Deklaration springen	gd ⚡ gD	

### Text suchen / Ersetzen (diese Befehle sind {b}-Bewegungsbefehle; bis auf :noh)

Wort unter Cursor suchen vorwärts ⚡ rückwärts	* ⚡ #	
Suchen/Ersetzen - Zeilenweise, einmal	:s/{t}/{t}/{n}	
Global	:s/{t}/{t}/g{n}	
Dokumentenweit, Global	:%s/{t}/{t}/g	
Nachfragen (geht immer mit gc)	:&s/{t}/{t}/gc	
Markierung der gefundenen Texte entfernen	:noh	
In mehreren Dateien suchen 1. {t} Suchbegriff, 2. {t} Dateien, Wildcards erlaubt	:vimgrep/{t}/{t}	
Zur nächsten Übereinstimmung springen	:cnext	
Liste der Übereinstimmungen öffnen ⚡ schließen	:copen ⚡ :cclose	

### Scrollen

Eine Zeile hoch ⚡ runter scrollen	Strg+E ⚡ Strg+Y	
Eine halbe Seite hoch ⚡ runter scrollen down/up	Strg+D ⚡ Strg+U	
Eine ganze Seite hoch ⚡ runter scrollen	Strg+F ⚡ Strg+B	
Text scrollen das Cursor oben ist ⚡ Erstes Zeichen	zt ⚡ z<Return>	
Text scrollen das Cursor zentriert ist ⚡ erst. Zeichen	zz ⚡ z.	
Text scrollen das Cursor unten ist ⚡ Erstes Zeichen	zb ⚡ z-	
Text scrollen das Zeile {n} oben/zent./unten ist	{n}zt ⚡ {n}zz ⚡ {n}zb	

## VIM – PROFIS

### Vervollständigungsmodus (Strg+X)

Im Einfügemodus: Vervollständigungsmodus öffnen	Strg+X ...	
Wortvervollständigung vorwärts } rückwärts	... Strg+n } Strg+p	
Danach Strg+n } Strg+p für nächstes } vorherigen		
Zeilenvervollständigung	... Strg+l	
Danach Strg+L ⌞ Strg+P } Strg+N für Vorherige } Nächstes		
Dateinamen vervollständigung	... Strg+f	
Danach Strg+f ⌞ Strg+n } Strg+p für Nächstes } Vorheriges		
Wörterbuchvervollständigung	... Strg+k	
Danach Strg+k ⌞ Strg+n } Strg+p für Nächstes } Vorherige		
Einfach in z.B. ~\vim\dict (Verzeichnis wenn nötig erstellen) eine Textdatei erstellen z.B. woerter		
Diese dann mit set dictionary+=~\vim\dict\woerter hinzufügen.		
Es können beliebig viele eigene Wörterbücher hinzugefügt werden.		
Die Konfiguration (set ...) kann auch in der ~\_vimrc bzw. ~\vimrc eingetragen werden und steht dann dauerhaft zur Verfügung.		

### Positionsmarkierungen / Sprünge

Zeigt alle Markierungen	:marks	
Markierung setzen	m{z}	
Zur Markierung springen	'{z} } `{z}	
Zur vorherigen Position springen	' '	
Zur vorherigen Position / letzte Arbeit springen	' "	?
Zur vorherigen Position / letzte Änderung springen	' .	
An zuletzt geöffneten Datei und dessen letzter Cursorposition springen. 0 = letzte, ..., 9 = neunt letzte	'0 } '1 } ... '9	?
Alle Sprünge auflisten	:jumps	
Zur vorherigen } nächsten Sprungposition springen	Strg+i } Strg+o	
Alle Änderung auflisten	:changes	
Zur nächsten } vorherigen Änderungspos. springen	g, } g;	

### Faltung (Folding)

Markierten Bereich falten } Falten bis {b}	zf } zf{b}	
Faltung unter Cursor umschalten	za	
Faltung unter Cursor öffnen } schließen	zo } zc	
Alle Faltung um eine Ebene öffnen } schließen	zr } zm	
Alle Faltungen öffnen } schließen	zR } zM	
Aktuelle Faltung löschen } Alle Faltungen löschen	zd } zE	

### Makros

Aufzeichnung starten	q{z}	
Aufzeichnung beenden	q	
Makro abspielen } 10 mal abspielen	@{z} } 10@{z}	
Letztes Makro erneut abspielen	@@	

### Tabs


Neuer Tab	:tabnew [{file}]	
Nächster Tab } vorheriger Tab	gt } gT	
Zum Tab Nr. {n} springen	{n}gt	
Tab an Position {n} verschieben	:tabmove {n}	
Aktuellen Tab schließen	:tabclose	
Alle Tabs bis auf den aktuellen schließen	:tabonly	
Befehl für alle Tabs z.B. :tabdo w speichert alle Tabs.	:tabdo {Befehl}	

### Fenster / Teilung / Puffer

Neue Teilung horizontal } vertikal	:new } :vnew	
In Fenster/Teilung/Tab Datei öffnen	:edit {Dateiname}	
Falls es bereits einen Puffer gibt wird dieser nun nicht mehr angezeigt.		
Datei in horizontaler } vertikaler Teilung öffnen	:split {File} :vsplit {File}	
Zum nächsten } vorherigen Puffer springen	:bnext :bprevious	
Damit lassen sich mehrere Puffer in einem Fenster verwalten. → :edit		
Puffer löschen (Datei schließen)	:bdelete	
Zum Puffer {n} springen	:buffer {n}	
Zum Puffer {filename} springen	:buffer {filename}	
Alle Puffer auflisten	:ls ⌞ :buffers	
Alle Puffer in Vertikale Teilungen	:vertical ball	
Alle Puffer in eigene Tabs verschieben	:tab ball	
Aktuelles Fenster in eigenen Tab verschieben	Strg+wT	
Fenster horizontal } vertikal teilen	Strg+ws } Strg+wv	
Fenster wechseln	Strg+ww	
Gezielt mit Strg+w und dann h,j,k,l für die Richtung		
Fenster schließen	Strg+wq	
Aktuelles Fenster mit dem nächsten tauschen	Strg+wx	
Alle Fenster gleich groß	Strg+w=	
Fenster nach h,j,k,l verschieben	Strg+wh } Strg+wJ	
Damit wird die Ausrichtung verändert.	Strg+wK } Strg+wL	



**Nützliche Befehle und Mappings:**

Tabulatoren neu erstellen (aus Tab werden Spaces) Sehr nützlich wenn Quellcode aus z.B. einer Webseite eingefügt wird. Verwendet diese Tabs zum einrücken, dann kann mit einem :retab alle Tabs in Spaces umgewandelt werden.	:retab	
Mappt die Strg+# Taste auf die in Vim/Neovim oft verwendete Strg-\ Taste Strg+\ kann bei deutscher Tastaturbelegung nicht verwendet werden! Mit diesem Mapping entspricht Strg-# dann Strg-\	:map <C-#> <C-\>	

**NeoVim Unterschiede:**

Terminal :term

Öffnet das Terminal NICHT in einem Split-Window. Man muss es vorher selbst splitten (: new). Befehle können erst nach betreten des Insert-Modus (I) eingegeben werden. Mit Strg-\ (Strg-# siehe oben) und Strg-N kann der Insert-Modus verlassen und mit Strg-W zwischen den Fenstern gewechselt werden.

\_vimrc-File

Liegt bei Neovim in C:\Users\<Username>\AppData\Local\nvim\init.vim

Ggfs. muss der nvim-Ordner noch erstellt werden.

In ...\Local\Colors liegen dann die Farben.

Die Zeile der .viminfo-Datei muss geändert werden (z.B. in .nviminfo).

Ansonsten kann ein \_vimrc für das init.vim verwendet werden.

**VIM Strg-Befehle in Visual Studio Code nutzen**

F1 → Preferences: Open Settings (JSON) und die folgenden Tasten:

<C-d>, <C-u>, <C-a>, <C-x>, <C-r>, <C-e>, <C-y>, <C-f>, <C-b> mit  
"vim.handleKeys": { "<C-d>": true, "<C-u>": true, ... }

Konfigurieren. Danach funktionieren in VS-Code diese Strg-Befehl nicht mehr. Stattdessen stehen die entsprechenden VIM-Funktionalitäten zur Verfügung.