The Pragmatic Programmers

Deutsche Übersetzung von O'REILLY®

Vim

in der Praxis

Texte bearbeiten im Handumdrehen

Drew Neil

Vorwort von Tim Pope

Übersetzung von Lars Schulten

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

Inhaltsverzeichnis

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

Teil V – Muster

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

KAPITEL 4

Visueller Modus

Vims visueller Modus ermöglicht es uns, eine Textauswahl zu definieren und mit ihr zu arbeiten. Das sollte ziemlich verständlich sein, da es das Modell ist, das die meisten Textverarbeitungsprogramme nutzen. Aber Vims Herangehensweise ist entschieden anders. Deswegen werden wir zunächst dafür sorgen, dass wir den visuellen Modus tatsächlich kapieren (Tipp 20, Seite 42).

Vim bietet drei Varianten des visuellen Modus, die mit Zeichen, Zeilen oder rechteckigen Textblöcken arbeiten. Wir werden uns Möglichkeiten ansehen, zwischen diesen Modi zu wechseln, und werden uns einige praktische Tricks für die Veränderung der Grenzen einer Auswahl aneignen (Tipp 21, Seite 44).

Wir werden lernen, dass der Punktbefehl genutzt werden kann, um Befehle des visuellen Modus zu wiederholen, und wir werden dabei sehen, dass er besonders effektiv ist, wenn man mit zeilenbasierten Regionen arbeitet. Wenn man mit einer zeichenbasierten Auswahl arbeitet, kann es geschehen, dass der Punktbefehl die eigenen Erwartungen nicht erfüllt. Wir werden sehen, dass man unter diesen Umständen Operatorbefehle eventuell vorziehen sollte.

Der visuelle Block-Modus ist in der Hinsicht besonders, dass er es uns ermöglicht, auf rechteckigen Textspalten zu operieren. Sie werden viele Anwendungsmöglichkeiten für diese Einrichtung finden, aber wir werden uns auf drei Tipps beschränken, die einige seiner Fähigkeiten illustrieren.

Tipp 20

Den visuellen Modus verstehen

Der visuelle Modus ermöglicht es uns, einen Textbereich auszuwählen und zu bearbeiten. Auch wenn das vollkommen selbstverständlich scheinen mag, unterscheidet sich Vims Vorstellung von der Auswahl von Text erheblich von der anderer Texteditoren.

Nehmen wir für einen Augenblick an, dass wir nicht mit Vim arbeiten, sondern ein Textfeld auf einer Webseite ausfüllen. Wir haben das Wort »März« geschrieben, wollten eigentlich aber »April« schreiben. Wir nutzen also die Maus und klicken doppelt auf das Wort, um es auszuwählen. Nachdem wir das Wort ausgewählt haben, können wir auf die Backspace-Taste tippen und dann als Ersatz den richtigen Monat eingeben.

Wahrscheinlich wissen Sie bereits, dass Sie die Backspace-Taste in diesem Fall gar nicht betätigen müssen. Wenn das Wort »März« ausgewählt ist, müssen wir nur den Buchstaben »A« eingeben. Das würde die Auswahl ersetzen, und wir könnten den Rest des Wortes »April« eingeben. Das ist nicht viel, aber eine gesparte Tastenbetätigung ist eine gewonnene Tastenbetätigung.

Wenn Sie erwarten sollten, dass dieses Verhalten auch für den visuellen Modus von Vim gilt, müssen Sie sich auf eine Überraschung gefasst machen. Einen Hinweis gibt Ihnen bereits der Name: Der visuelle *Modus* ist einfach ein anderer Modus, und das bedeutet, dass jede Taste eine andere Funktion hat.

Viele der Befehle, die ihnen aus dem normalen Modus vertraut sind, funktionieren im visuellen Modus auf genau die gleiche Weise. Wir können weiterhin `h`, `j`, `k` und `l` als Cursortasten nutzen. Wir können `f{Zeichen}` nutzen, um zu einem Zeichen auf der aktuellen Zeile zu springen, und den Sprung dann mit den Befehlen `;` und `,` wiederholen beziehungsweise umkehren. Wir können sogar den Suchbefehl (und `n`/`N`) nutzen, um auf Basis von Mustervergleichen zu springen. Jedes Mal, wenn wir im visuellen Modus den Cursor bewegen, ändern wir die Grenzen der Auswahl.

Einige der Befehle des visuellen Modus erfüllen die gleichen Grundfunktionen wie im normalen Modus, weisen aber eine leichte Abweichung auf. Beispielsweise löscht der Befehl `c` in beiden Modi den angegebenen Text und wechselt in den Einfügemodus. Der Unterschied besteht darin, wie wir den Bereich angeben, der bearbeitet werden soll. Im normalen Modus geben wir erst den Befehl ein und definieren den Bereich dann über eine Bewegung.

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

Er wird, wie Sie sich aus Tipp 12, Seite 27 erinnern werden, als Operatorbefehl bezeichnet. Im visuellen Modus beginnen wir hingegen damit, dass wir eine Auswahl machen und dann den Verändernbefehl anstoßen. Diese Umkehrung der Abfolge gilt allgemein für alle Operatorbefehle (siehe Tabelle 2, *Vims Operatorbefehle*, Seite 28). Den meisten scheint der Ansatz des visuellen Modus intuitiver.

Kehren wir zu dem Beispiel zurück, in dem wir das Wort »März« in das Wort »April« ändern wollen. Nehmen wir diesmal an, dass wir die Schranken einer Webseite hinter uns gelassen haben und uns wieder im behaglichen Vim-Umfeld befinden. Wir setzen unseren Cursor irgendwo auf das Wort »März« und nutzen `viw`, um das Wort visuell auszuwählen. Jetzt können wir allerdings nicht einfach das Wort »April« eingeben, weil das den Befehl `A` in Gang setzen und den Text »pril« anhängen würde! Stattdessen werden wir den Befehl `c` nutzen, um die Auswahl zu ändern, das Wort zu löschen und in den Einfügemodus zu wechseln, in dem wir dann das Wort »April« vollständig eingeben können. Dieses Verwendungsmuster ähnelt dem in unserem ersten Beispiel, nur dass wir die `c`-Taste statt der Backspace-Taste nutzen.

Der Auswahlmodus

In einer gängigen Textverarbeitungsumgebung wird der ausgewählte Text gelöscht, wenn ein druckbares Zeichen eingegeben wird. Der visuelle Modus von Vim folgt dieser Konvention nicht, sein Auswahlmodus hingegen schon. Laut Vims eingebauter Dokumentation, ähnelt er dem Auswahlmodus von Microsoft Windows (siehe `:h Select-mode` ⓘ). Druckbare Zeichen bewirken, dass die Auswahl gelöscht, in den Einfügemodus gewechselt und dann das getippte Zeichen eingegeben wird.

Zwischen dem visuellen Modus und dem Auswahlmodus kann mit `<C-g>` gewechselt werden. Der einzige sichtbare Unterschied ist die Meldung am unteren Bildschirmrand, die von `-- VISUAL --` zu `-- SELECT --` wechselt und umgekehrt. Aber wenn wir im Auswahlmodus ein druckbares Zeichen eingeben, ersetzt es die Auswahl, und es wird in den Einfügemodus gewechselt. Natürlich können Sie auch einfach im visuellen Modus die `c`-Taste nutzen, um die Auswahl zu ändern.

Wenn Ihnen die modale Natur von Vim liegt, werden Sie wenig Gefallen am Auswahlmodus finden, der Benutzern die Hand reicht, die sich wünschen, Vim würde sich etwas mehr wie andere Texteditoren verhalten. Mir fällt nur eine Gelegenheit ein, bei der ich stets den Auswahlmodus nutze: Wenn ich ein Plugin nutze, das die Snippet-Einrichtung von TextMate emuliert, denn dann markiert der Auswahlmodus den aktiven Platzhalter.

Tipp 21

Eine visuelle Auswahl definieren

Die drei Submodi des visuellen Modus befassen sich mit unterschiedlichen Arten von Text. In diesem Tipp werden wir uns die Möglichkeiten ansehen, um die verschiedenen visuellen Submodi zu aktivieren und zwischen ihnen umzuschalten.

Vim hat drei unterschiedliche visuelle Modi. Im *zeichenbasierten visuellen Modus* können wir alles von einem einzelnen Zeichen bis zu einem Zeichenbereich in einer Zeile oder über mehrere Zeilen hinweg auswählen. Er ist für die Bearbeitung von einzelnen Wörtern oder Satzfragmenten geeignet. Wenn wir vollständige Zeilen bearbeiten wollen, können wir stattdessen den *zeilenbasierten* visuellen Modus nutzen. Schließlich ermöglicht es uns der *blockbasierte* visuelle Modus, mit Spaltenbereichen im Dokument zu arbeiten. Der blockbasierte Modus ist recht speziell und wird deswegen ausführlicher in Tipp 24, Seite 50, Tipp 25, Seite 52 und Tipp 26, Seite 54 behandelt.

Visuelle Modi aktivieren

Der Befehl `v` ist unser Portal für den visuellen Modus. Im normalen Modus können wir `v` alleine drücken, um den zeichenbasierten visuellen Modus zu aktivieren. Der zeilenbasierte visuelle Modus wird mit `V` (mit der Hochstelltaste) und der blockbasierte visuelle Modus mit `<C-v>` (mit der Strg-Taste) aktiviert. Diese Befehle werden von der folgenden Tabelle zusammengefasst:

Befehl	Effekt
`v`	Zeichenbasierten visuellen Modus aktivieren
`V`	Zeilenbasierten visuellen Modus aktivieren
`<C-v>`	Blockbasierten visuellen Modus aktivieren
`gv`	Den zuletzt im visuellen Modus ausgewählten Bereich wieder auswählen

Der Befehl `gv` ist eine nützliche kleine Abkürzung. Er wählt den Textbereich wieder aus, der zuletzt in einem visuellen Modus ausgewählt war. Dieser Befehl verhält sich immer gleich, ganz gleich ob die letzte Auswahl im zeichenbasierten, zeilenbasierten oder blockbasierten Modus erfolgte. Er könnte nur dann verwirrt werden, wenn die letzte Auswahl seitdem gelöscht wurde.

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

Zwischen visuellen Modi wechseln

Wir können zwischen den verschiedenen Spielarten des visuellen Modus auf die gleiche Weise wechseln, wie wir sie aus dem normalen Modus aktivieren. Wenn wir im zeichenbasierten Modus sind, können wir in die zeilenbasierte Spielart wechseln, indem wir `V` drücken, oder in die blockbasierte Spielart, indem wir `<C-v>` drücken. Aber wenn wir im zeichenbasierten visuellen Modus `v` drücken, wechseln wir wieder in den normalen Modus zurück. Sie können die `v`-Taste also als einen Schalter zwischen dem normalen Modus und dem zeichenbasierten visuellen Modus betrachten. Die Tastenbetätigungen `V` und `<C-v>` schalten ebenfalls zwischen dem normalen Modus und der jeweiligen Spielart des visuellen Modus um. Natürlich können Sie auch jederzeit `<Esc>` oder `<C-[>` drücken, um wieder in den normalen Modus zurückzugelangen (genau so, wie Sie auch den Einfügemodus verlassen). Die folgende Tabelle fasst die Befehle für den Wechsel zwischen den visuellen Modi zusammen:

Befehl	Effekt
`<Esc>` / `<C-[>`	In den normalen Modus wechseln
`v` / `V` / `<C-v>`	In den normalen Modus wechseln (bei Verwendung im zeichenbasierten, zeilenbasierten respektive blockbasierten visuellen Modus)
`v`	In den zeichenbasierten visuellen Modus wechseln
`V`	In den zeilenbasierten visuellen Modus wechseln
`<C-v>`	In den blockbasierten visuellen Modus wechseln
`o`	Zum anderen Ende des markierten Texts gehen

Das offene Ende einer Auswahl umschalten

Der Bereich einer Auswahl im visuellen Modus wird durch zwei Enden definiert: Das eine Ende ist fest, das andere ist offen und bewegt sich frei mit unserem Cursor. Wir können den Befehl `o` nutzen, um das offene Ende umzuschalten. Das ist äußerst praktisch, wenn man bei der Definition einer Auswahl auf halbem Wege feststellt, dass man an der falschen Stelle begonnen hat. Man muss nicht den visuellen Modus verlassen und neu beginnen, sondern kann einfach `o` drücken und die Grenzen der Auswahl neu definieren. Das folgende Beispiel illustriert den Einsatz dieser Technik:

Tastenaktionen	Pufferinhalt
{Ausgangszustand}	Von hier nach dort auswählen.
vbb	Von hier nach dort auswählen.
o	Von hier nach dort auswählen.
e	Von hier nach dort auswählen.

Tipp 22

Zeilenbasierte visuelle Befehle wiederholen

Wenn wir den Punktbefehl nutzen, um eine Änderung zu wiederholen, die im visuellen Modus erfolgt ist, wird diese Änderung auf dem gleichen Textbereich wiederholt. In diesem Tipp werden wir eine Änderung an einer zeilenbasierten Auswahl vornehmen und diese dann mit dem Punktbefehl wiederholen.

Wenn wir im visuellen Modus einen Befehl ausführen, werden wir wieder in den normalen Modus zurückversetzt, und die Auswahl des Textes, der im visuellen Modus markiert wurde, wird aufgehoben. Was also sollen wir tun, wenn wir einen weiteren Befehl des visuellen Modus auf dem gleichen Textbereich ausführen wollen?

Angenommen, wir hätten den folgenden Auszug mit falsch formatiertem Python-Code:

visual_mode/fibonacci-malformed.py

```
def fib(n):
    a, b = 0, 1
    while a < n:
print a,
a, b = b, a+b
fib(42)
```

Dieser Beispielcode nutzt vier Leerzeichen pro Einzugsebene. Zunächst müssen wir Vim so konfigurieren, dass er diesen Stil übernimmt.

Vorbereitung

Wir sollten die Vim-Parameter `shiftwidth` und `softtabstop` auf 4 setzen und `expandtab` aktivieren, damit die Befehle `<` und `>` korrekt funktionieren. (Wenn Sie verstehen wollen, wie diese Parameter zusammenspielen, sollten

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

Sie sich die »Tabs and Spaces«-Episode auf *Vimcasts.org* ansehen.)[1] Das bewirkt der folgende Einzeiler:

```
⇒ :set shiftwidth=4 softtabstop=4 expandtab
```

Einmal einrücken, dann wiederholen

In unserem falsch formatierten Python-Beispiel sollten die beiden Zeilen unter dem Schlüsselwort `while` zwei Ebenen tiefer eingerückt werden. Das könnten wir reparieren, indem wir den Text auswählen und den `>`-Befehl nutzen, um ihn einzurücken. Doch das würde den Einzug nur um eine Ebene erhöhen und uns dann wieder in den normalen Modus zurückversetzen.

Eine Lösung wäre, den Text erneut mit dem Befehl `gv` auszuwählen und den Einzugsbefehl dann erneut aufzurufen. Aber wenn Sie sich bereits etwas in Vim eingelebt haben, sollten bei Ihnen jetzt die Alarmglocken schrillen.

Wenn wir etwas wiederholen müssen, ist der Punktbefehl unser bester Freund. Anstatt wieder den gleichen Textbereich auszuwählen und den gleichen Befehl manuell auszuführen, können wir einfach im normalen Modus die `.`-Taste betätigen. Hier sehen Sie das in Aktion:

Tastenaktionen	Pufferinhalt

`{Ausgangszustand}`	`def fib(n):`
	`  a, b = 0, 1`
	`  while a < n:`
	`print a,`
	`a, b = b, a+b`
	`fib(42)`

`Vj`	`def fib(n):`
	`  a, b = 0, 1`
	`  while a < n:`
	`print a,`
	`a, b = b, a+b`
	`fib(42)`

`>.`	`def fib(n):`
	`  a, b = 0, 1`
	`  while a < n:`
	`print a,`
	`a, b = b, a+b`
	`fib(42)`

Wenn Ihnen das Zählen im Blut liegt, ziehen Sie es vielleicht vor, das Ziel mit einem einzigen Streich zu erreichen, indem Sie im visuellen Modus `2>` ausführen. Ich ziehe die Verwendung des Punktbefehls vor, da er mir eine

1. http://vimcasts.org/e/2

unmittelbare visuelle Rückmeldung gibt. Wenn Sie den Einzugsbefehl erneut anstoßen müssen, müssen Sie einfach nur erneut `.` betätigen. Und wenn ich es übertreibe und über das Ziel hinausschieße, drücke ich einfach die `u`-Taste, um alles wieder gerade zu rücken. In Tipp 11, Seite 25 betrachten wir diese Frage etwas ausführlicher.

Wenn wir den Punktbefehl nutzen, um einen Befehl des visuellen Modus zu wiederholen, wirkt er sich auf eben die Textmenge aus, die bei der letzten Auswahl im visuellen Modus markiert wurde. Dieses Verhalten ist in der Regel praktisch, wenn unsere visuelle Auswahl zeilenbasiert ist, kann aber zu überraschenden Ergebnissen führen, wenn die Auswahl zeichenbasiert ist. Als Nächstes werden wir uns ein Beispiel ansehen, das das illustriert.

Tipp 23

Ziehen Sie Operatoren wenn möglich visuellen Befehlen vor

Der visuelle Modus mag zwar etwas intuitiver sein als der normale Operationsmodus von Vim, hat aber seine Schwächen: Er verträgt sich nicht immer so gut mit dem Punktbefehl. Diese Schwäche können wir umschiffen, wenn wir immer dann, wenn es angemessen ist, Operatoren des normalen Modus nutzen.

Angenommen, Sie wollen die Links der folgenden Liste so ändern, dass sie Großbuchstaben nutzen:

visual_mode/list-of-links.html

```
<a  href="#">gelb</a>
<a  href="#">blau</a>
<a  href="#">orange</a>
```

Wir können den inneren Inhalt eines Tags auswählen, indem wir `vit` nutzen. Das kann folgendermaßen gelesen werden: *visuell innerhalb* des *Tags* auswählen. Der Befehl `it` ist eine besondere Art der Bewegung, die als Textobjekt bezeichnet wird. Mit diesem werden wir uns in Tipp 51, Seite 136 genauer befassen.

Einen visuellen Operator nutzen

Im visuellen Modus erstellen wir eine Auswahl, auf der wir dann operieren. In diesem Fall könnten wird den Befehl `U` nutzen, der die ausgewählten Zeichen in Großbuchstaben umwandelt (`:h v_U` ⓘ). Sehen Sie sich dazu Tabelle 4 an.

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

Tabelle 4: Großbuchstaben im visuellen Modus

Tastenaktionen	Pufferinhalt
{Ausgangszustand}	gelb blau orange
`vit`	gelb blau orange
`U`	GELB blau orange

Nachdem wir die erste Zeile umgewandelt haben, wollen wir nun die gleiche Änderung auf den nächsten beiden Zeilen durchführen. Wie wäre es, wenn wir dazu die Punktformel nutzten?

`j.` rückt den Cursor in die nächste Zeile vor und wiederholt die letzte Änderung. Bei der zweiten Zeile funktioniert das wunderbar, aber wenn wir es erneut probieren, erhalten wir das folgende seltsame Ergebnis:

```
<a href="#">GELB</a>
<a href="#">BLAU</a>
<a href="#">ORANge</a>
```

Verstehen Sie, was passiert ist? Wenn der Befehl des visuellen Modus wiederholt wird, wirkt er sich auf den gleichen Textbereich aus (siehe `:h visual-repeat` ⓘ). Hier hat der ursprüngliche Befehl ein Wort geändert, das aus vier Buchstaben besteht. Bei Zeile zwei funktioniert das wunderbar, da auch diese ein Wort mit vier Buchstaben enthält, aber es schlägt fehl, wenn wir versuchen, den Befehl für ein Wort zu wiederholen, das aus sechs Buchstaben besteht.

Einen Operator des normalen Modus verwenden

Der `U`-Befehl des visuellen Modus hat ein Äquivalent im normalen Modus: `gU{Bewegung}` (`:h gU` ⓘ). Wenn wir diesen nutzen, um die erste Änderung durchzuführen, können wir die nachfolgenden Bearbeitungsschritte mit der Punktformel durchführen, wie Sie in Tabelle 5, *Normaler Operator im visuellen Modus*, Seite 50 sehen.

Erörterung

Beide Techniken verlangen nur vier Tastenaktionen: `vitU` bzw. `gUit`, aber die elementare Funktionsweise ist ganz anders. Die Tastenbetätigungen im

Tabelle 5: Normaler Operator im visuellen Modus

Tastenaktionen	Pufferinhalt
{Ausgangszustand}	gelb blau orange
gUit	GELB blau orange
j.	GELB BLAU orange
j.	GELB BLAU ORANGE

visuellen Modus können als zwei eigenständige Befehle betrachtet werden: `vit` trifft die Auswahl, und `U` transformiert diese. In Gegensatz dazu kann `gUit` als ein einziger Befehl betrachtet werden, der aus einem Operator (`gU`) und einer Bewegung (`it`) besteht.

Wenn wir den Punktbefehl so einrichten, dass er etwas Nützliches wiederholt, dann sind wir besser bedient, wenn wir den visuellen Modus nicht nutzen. Allgemein könnte man die Regel formulieren, dass man Operatorbefehle ihren Äquivalenten im visuellen Modus vorziehen sollte, wenn man repetitive Änderungen vornehmen muss.

Das soll nicht heißen, dass der visuelle Modus tabu ist. Er hat trotzdem seine Berechtigung. Es müssen nicht alle Bearbeitungsaufgaben wiederholt werden. Der visuelle Modus ist für einzelne Änderungen vollkommen angemessen. Und obgleich Vims Bewegungen eine hohe Präzision bieten, müssen wir manchmal einen Textbereich verändern, dessen Struktur schwer zu durchschauen ist. In einem solchen Fall ist der visuelle Modus das richtige Werkzeug für die Aufgabe.

Tipp 24

Tabellarische Daten mit dem blockbasierten Modus bearbeiten

Mit Textzeilen können wir in jedem Texteditor arbeiten, aber die Arbeit mit Textspalten erfordern ein spezialisierteres Werkzeug. Vim bietet entsprechen-

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

de Einrichtungen in Form des blockbasierten visuellen Modus. Diese werden wir nutzen, um eine Tabelle in reiner Textform zu bearbeiten.

Angenommen, wir haben eine Tabelle in einfacher Textform wie die folgende:

visual_mode/chapter-table.txt

```
Kapitel                 Seite
normaler Modus            15
Einfügemodus              31
visueller Modus           44
```

Wir möchten eine vertikale Linie aus senkrechten Strichen zeichnen, die die beiden Spalten trennt und dem Ganzen ein tabellenmäßigeres Aussehen verleiht. Aber zunächst reduzieren wir den Abstand zwischen den beiden Spalten, die weiter voneinander entfernt sind, als es erforderlich wäre. Beide Änderungen können wir im visuellen Blockmodus vornehmen. Wie das geht, sehen Sie in Tabelle 6, *Einen Trennstrich zwischen Spalten einfügen*, Seite 52.

Zunächst nutzen wir `<C-v>`, um den visuellen Blockmodus zu aktivieren. Dann definieren wir die Spalte, indem wir unseren Cursor mehrere Zeilen nach unten bewegen. `x` löscht diese Spalte, und der Punktbefehl wiederholt diesen Befehl für den gleichen Textbereich. Wir wiederholen die Operation so lange, bis die beiden Spalten einen angemessenen Abstand haben.

Wir hätten auf den Punktbefehl verzichten und stattdessen unsere Spaltenauswahl erweitern können, indem wir den Cursor zwei oder drei Schritte nach rechts verschoben hätten. Dann hätte eine einzige Löschoperation gereicht. Ich ziehe das unmittelbare visuelle Feedback vor, das ich erhalte, wenn ich einzelne Spalten lösche und das wiederhole.

Nachdem wir die beiden Textspalten so gestaltet haben, wie wir sie haben wollen, können wir die Trennlinie zwischen ihnen zeichnen. Dazu wählen wir einfach unsere letzte visuelle Auswahl mit dem Befehl `gv` erneut aus und nutzen dann `r|`, um alle Zeichen in der Auswahl durch einen senkrechten Strich zu ersetzen.

Wo wir gerade dabei sind, können wir auch gleich noch eine horizontale Linie einzeichnen, die die Spaltenüberschriften von den Zeilen darunter trennt. Dazu nutzen wir eine fixe Zeilenkopie, um die oberste Zeile zu duplizieren (`yyp`), und ersetzen dann jedes Zeilen in dieser Zeile durch ein Minuszeichen (`Vr-`).

Tabelle 6: Einen Trennstrich zwischen Spalten einfügen

Tastenaktionen	Pufferinhalt
{Ausgangszustand}	Kapitel ▉ Seite normaler Modus 15 Einfügemodus 31 visueller Modus 44
<C-v>3j	Kapitel ▉ Seite normaler Modus ▉ 15 Einfügemodus ▉ 31 visueller Modus ▉ 44
x...	Kapitel ▉ Seite normaler Modus 15 Einfügemodus 31 visueller Modus 44
gv	Kapitel ▉ Seite normaler Modus ▉ 15 Einfügemodus ▉ 31 visueller Modus ▉ 44
r|	Kapitel ▉ Seite normaler Modus | 15 Einfügemodus | 31 visueller Modus | 44
yyp	Kapitel | Seite Kapitel | Seite normaler Modus | 15 Einfügemodus | 31 visueller Modus | 44
Vr-	Kapitel | Seite ----------------------- normaler Modus | 15 Einfügemodus | 31 visueller Modus | 44

Tipp 25

Textspalten ändern

Wir können den visuellen Blockmodus nutzen, um simultan Text in mehrere Textzeilen einzufügen.

Der visuelle Blockmodus ist nicht nur nützlich, wenn man mit tabellarischen Daten arbeitet. Häufig kann uns diese Einrichtung auch nutzen, wenn wir

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

mit Code arbeiten. Schauen Sie sich beispielsweise das folgende (suboptimale) CSS-Fragment an:

visual_mode/sprite.css

```
li.one   a{ background-image: url('/images/sprite.png'); }
li.two   a{ background-image: url('/images/sprite.png'); }
li.three a{ background-image: url('/images/sprite.png'); }
```

Nehmen wir an, dass die Datei `sprite.png` aus dem Verzeichnis `images/` in ein Verzeichnis namens `components/` verschoben wurde. Wir müssen alle Zeilen so ändern, dass sie den neuen Ort dieser Datei referenzieren. Das könnten wir im visuellen Blockmodus so tun, wie Sie es in Tabelle 7 sehen.

Tabelle 7: In mehrere Zeilen einfügen

Tastenaktionen	Pufferinhalt
{Ausgangszustand} *normaler Modus*	li.one a{ background-image: url('/images/sprite.png'); } li.two a{ background-image: url('/images/sprite.png'); } li.three a{ background-image: url('/images/sprite.png'); }
`<C-v>jje` *visueller Modus*	li.one a{ background-image: url('/images/sprite.png'); } li.two a{ background-image: url('/images/sprite.png'); } li.three a{ background-image: url('/images/sprite.png'); }
`c` *Einfügemodus*	li.one a{ background-image: url('//sprite.png'); } li.two a{ background-image: url('//sprite.png'); } li.three a{ background-image: url('//sprite.png'); }
components *Einfügemodus*	li.one a{ background-image: url('/components//sprite.png'); } li.two a{ background-image: url('//sprite.png'); } li.three a{ background-image: url('//sprite.png'); }
`<Esc>` *normaler Modus*	li.one a{ background-image: url('/components/sprite.png'); } li.two a{ background-image: url('/components/sprite.png'); } li.three a{ background-image: url('/components/sprite.png'); }

Die Vorgehensweise sollte Ihnen mittlerweile recht vertraut sein. Zunächst definieren wir die Auswahl, mit der wir arbeiten wollen. Diese stellt einen rechteckigen visuellen Block dar. Wenn wir die Taste `c` betätigen, verschwindet der gesamte ausgewählte Text und wir wechseln in den Einfügemodus.

Wenn wir im Einfügemodus das Wort »components« eingeben, erscheint es nur auf der obersten Zeile. In den beiden Zeilen darunter geschieht nichts.

Wir sehen den eingegebenen Text erst, wenn wir `<Esc>` drücken, um in den normalen Modus zurückzukehren.

Das Verhalten des Vim-Befehls, der visuelle Blockänderungen durchführt, könnte etwas überraschend sein. Es wirkt inkonsistent, dass sich die Löschung auf alle markierten Zeilen zugleich auswirkt, während sich das Einfügen nur auf die oberste Zeile auswirkt (zumindest so lange, wie wir uns im Einfügemodus befinden). Manche Texteditoren bieten ähnliche Einrichtungen, aktualisieren aber alle ausgewählten Zeilen gleichzeitig. Wenn Sie an derartiges Verhalten gewöhnt sind (was bei mir der Fall war), dann könnte Ihnen Vims Implementierung weniger ausgereift erscheinen. Aber in der Praxis ist das Endergebnis das Gleiche. Solange Sie immer nur kurze Zeit in den Einfügemodus wechseln, sollte es keine Überraschungen geben.

Tipp 26

Etwas ans Ende der Zeilen eines unregelmäßigen visuellen Blocks anhängen

Der visuelle Blockmodus ist großartig, wenn man mit rechteckigen Textbereichen wie Spalten und Zeilen arbeiten muss, ist aber dennoch nicht auf rechteckige Textbereiche beschränkt.

Das folgende JavaScript-Fragment ist uns bereits begegnet:

the_vim_way/2_foo_bar.js

```
var foo = 1
var bar = 'a'
var foobar = foo + bar
```

Es handelt sich um drei aufeinanderfolgende Zeilen jeweils unterschiedlicher Länge. Wir möchten ein Semikolon ans Ende jeder Zeile anhängen. In Tipp 2, Seite 4 haben wir dieses Problem mit dem Punktbefehl gelöst, aber wir hätten ebenso gut den visuellen Blockmodus nutzen können. Tabelle 8, *Im visuellen Blockmodus ein Semikolon an mehrere Zeilen anhängen*, Seite 55 zeigt, wie das geht.

Nachdem wir den visuellen Blockmodus aktiviert haben, erweitern wir unsere Auswahl bis zum Ende aller Zeilen, indem wir `$` drücken. Auf den ersten Blick könnte man erwarten, dass das zu Problemen führt, da alle Zeilen eine unterschiedliche Länge haben. Aber in diesem Kontext versteht Vim, dass wir unsere Auswahl jeweils bis ans Ende aller ausgewählten Zeilen erweitern wollen. So können wir aus den Schranken der rechteckigen Auswahl ausbrechen und eine Auswahl erstellen, die die unregelmäßigen Enden unseres Textes nachzeichnet.

Drew Neil, Vim in der Praxis, O´Reilly, ISBN 9783955615789

Tabelle 8: Im visuellen Blockmodus ein Semikolon an mehrere Zeilen anhängen

Tastenaktionen	Pufferinhalt

{Ausgangszustand}
normaler Modus
	`var foo = 1`
	`var bar = 'a'`
	`var foobar = foo + bar`

`<C-v>jj$`
visueller Block
	`var foo = 1`
	`var bar = 'a'`
	`var foobar = foo + bar`

`A ;`
Einfügemodus
	`var foo = 1;`
	`var bar = 'a'`
	`var foobar = foo + bar`

<Esc>
normaler Modus
	`var foo = 1;`
	`var bar = 'a';`
	`var foobar = foo + bar;`

Nachdem wir unsere Auswahl definiert haben, können wir an alle Zeilen Text anhängen, indem wir den Befehl `A` nutzen (siehe den Kasten *Vims Konventionen für die Tasten »i« und »a«* unten). Das bringt uns auf der obersten Zeile unserer Auswahl in den Einfügemodus. Alles, was wir nun eingeben, erscheint während der Eingabe nur in dieser Zeile, aber sobald wir wieder in den normalen Modus zurückkehren, werden unsere Änderungen in die anderen Zeilen übernommen, die wir ausgewählt hatten.

Vims Konventionen für die Tasten »i« und »a«

Vim bietet einige Konventionen für den Wechsel vom normalen Modus in den Einfügemodus. Die Befehle `i` und `a` wechseln in den Einfügemodus und positionieren den Cursor vor beziehungsweise hinter dem aktuellen *Zeichen*. Die Befehle `I` und `A` verhalten sich ähnlich, positionieren den Cursor aber am Anfang beziehungsweise am Ende der aktuellen *Zeile*.

Vim hat ähnliche Konventionen für den Wechsel vom visuellen Blockmodus in den Einfügemodus. Die Befehle `I` und `A` wechseln in den Einfügemodus und platzieren den Cursor am Anfang beziehungsweise am Ende der *Auswahl*. Was also ist mit den Befehlen `i` und `a`; was tun diese im visuellen Modus?

Im visuellen Modus und im Operator-ausstehend-Modus haben die Befehle `i` und `a` eine andere Semantik: Sie bilden die erste Hälfte eines *Textobjekts*. Textobjekte werden in Tipp 51, Seite 136 ausführlicher behandelt. Wenn Sie im visuellen Blockmodus eine Auswahl gemacht haben und sich wundern, dass ein `i` Sie nicht in den Einfügemodus bringt, sollten Sie es stattdessen einfach mit `I` probieren.