Microsoft

Visual Studio

Visual Studio Code Tipps & Tricks Vol. 1

1. Auflage – März 2016, 1. Aktualisierung im April 2016
© Microsoft Deutschland GmbH 2016

Microsoft Deutschland GmbH · Konrad-Zuse-Str. 1 · 85716 Unterschleißheim
Tel: 089 3176 0 · www.microsoft.de · www.techwiese.de

Autoren: Tobias Kahlert und Kay Giza · Microsoft Deutschland GmbH
Redaktion: Mathias Schiffer

Alle Rechte vorbehalten. Dieses Dokument dient lediglich Informationszwecken. Irrtümer und Fehler vorbehalten. **Dieses Dokument immer aktuell unter http://aka.ms/VSCodeTippsTricks**

Inhalt

Visual Studio Code? 4
Vorwort 5
Was ist Visual Studio Code? 6
1. Tipp – Updates vor allen anderen erhalten als VS Code Insider 7
2. Tipp – Mehrere Eingabemarken 8
3. Tipp – Visual Studio Code mit der Kommandopalette steuern 8
4. Tipp – Sprache für eine Datei auswählen 9
5. Tipp – AutoSave 10
6. Tipp – Unwichtige Ordner verstecken 10
7. Tipp – Suchergebnisse ausblenden 11
8. Tipp – RegEx-Treffer beim Ersetzen wieder einsetzen 12
9. Tipp – Emmet Snippets 12
10. Tipp – Tastenkürzel-Akkorde 13
11. Tipp – Markdown-Vorschau 14
12. Tipp – CSS-Selektoren visualisieren 14
13. Tipp – „Latest Version“ von Abhängigkeiten in package.json 15
14. Tipp – Git Quick Change Info 16
15. Tipp – Git-Ansicht aktualisieren 17
16. Tipp – Git-Inline-Vergleich 17
17. Tipp – Dateien vergleichen 18
18. Tipp – Extensions 20
19. Tipp – Automatische Taskerkennung für Gulp, Grunt und Jake 20
20. Tipp – PHP-Konfiguration in Visual Studio Code 21
21. Tipp – Kommandozeilenparameter von Visual Studio Code 22
Konsolenunterstützung unter Linux 22
22. Tipp – Speicherorte in Visual Studio Code 22
Einstellungen, Tastenkombinationen und Code-Snippets 22
Extensions 23
23. Tipp – Einstellungen und Extensions synchronisieren 23
24. Tipp – Crash-Berichte ausschalten 23
25. Tipp – Übermittlung von Telemetrie-Daten abschalten 23

26. Tipp – Navigieren zwischen verwendeten Dateien 24
27. Tipp – Dateien in eigenem Editor-Fenster öffnen 24
28. Tipp – Tastenkombinationen anpassen 25
29. Tipp – Lieblings-Tastenkombinationen der Autoren 25
30. Tipp – Änderung der Oberflächensprache 26
31. Tipp – Zeilenumbruch im Editor 26

Visual Studio Code?

Mit Visual Studio Code bietet sich Entwicklern ein neues Entwicklungswerkzeug an, das die Schnörkellosigkeit und Geradlinigkeit eines Code-Editors mit dem Besten verbindet, das Entwickler für den Code-Build-Debug-Zyklus benötigen. Visual Studio Code ist der erste reine Code-Editor und das erste plattformübergreifende Entwicklungswerkzeug (mit Unterstützung von OS X, Linux und Windows) der Visual Studio-Familie.

Der Kern von Visual Studio Code (VS Code) liegt in einem leistungsstarken, schnellen Code-Editor, der ideal ist für den täglichen Gebrauch. Bereits die Beta-Version von Code wies viele der Features auf, die Entwickler von einem Code- und Text-Editor erwarten – einschließlich Navigation, Tastaturunterstützung furch anpassbare Bindungen, Syntax-Highlighting, Klammernzuordnung, automatische Einrückung und Snippets, und dies mitsamt Unterstützung für dutzende Sprachen.

Für anspruchsvolles Programmieren müssen Entwickler Quelltext oft umfassender bearbeiten als einfach nur Text. Visual Studio Code glänzt mit eingebauter Unterstützung für stets verfügbare IntelliSense-Vervollständigung, reichhaltigeres semantisches Quelltext-Verständnis und -Navigation sowie Refactoring. In der Preview-Version bietet Code erweiterte Unterstützung für ASP.NET Core-Entwicklung mit C# sowie Node.js-Entwicklung mit TypeScript und JavaScript auf Basis derselben Basistechnologien wie in Visual Studio.

Code umfasst großartige Tools für Web-Technologien wie HTML, CSS, Less, Sass und JSON. Code punktet auch mit Integration für Package-Manager und Repositories, Builds und weitere häufige Aufgaben, die die täglichen Abläufe beschleunigen. Außerdem versteht VS Code Git nicht nur, sondern liefert auch großartige Git-Workflows und Source-Diffs – im Editor integriert.

Doch Entwickler verbringen nicht ihre gesamte Zeit mit dem Schreiben von Code: Sie wechseln beständig zwischen Coding und Debugging hin und her. Debugging ist das populärste Feature in Visual Studio, und oftmals dasjenige IDE-Wunschfeature, das Entwickler für ein produktives Coding ersehnen. Visual Studio Code bietet eine geradlinige, integrierte Debugging-Funktionalität mit Unterstützung für Node.js-Debugging und weitere Zukunftspläne.

Architektonisch kombiniert Visual Studio Code das Beste aus Web-, nativen und sprachspezifischen Technologien. Mit der GitHub Electron Shell führt Code Web-Technologien wie JavaScript und Node.js mit der Geschwindigkeit und Flexibilität nativer Apps zusammen. Code nutzt eine neuere, schnellere Version desselben industriebewährten HTML-basierenden Editors, der dem Cloud-Editor „Monaco“, den F12-Tools des Internet Explorers und weiteren Projekten zugrunde liegt.

Zudem nutzt Code eine Architektur, die ihm viele derselben Technologien zugänglich macht, die Visual Studio einsetzt, inkl. Roslyn für .NET, TypeScript, die Visual Studio Debugging Engine und noch weitere. Visual Studio bietet ein offengelegtes Erweiterungsmodell, mit dem Entwickler Plug-Ins entwickeln und nutzen sowie ihre Edit-Build-Debug-Erfahrung umfassend anpassen können.

Wenn Sie Editor-zentrische Entwicklungswerkzeuge bevorzugen oder Web- und Cloud-Anwendungen plattformübergreifend entwickeln, laden wir Sie herzlich ein, die bereits vorliegenden Versionen von Visual Studio Code zu testen und uns Ihr Feedback zukommen zu lassen!

Vorwort

Wir haben vor etwas weniger als fünf Jahren mit der Entwicklung von Komponenten für das heutige Visual Studio Code begonnen. Uns war von Anfang an bewusst; wenn wir ein Werkzeug für Entwickler bauen, dann müssen wir es auch selbst nutzen.

Tatsächlich haben wir bereits nach nur vier Monaten ausschließlich mit unserem Vorgänger von Code gearbeitet. Dies, obwohl es am Anfang viele Limitierungen gab, und es ziemlich holprig war. Zum Beispiel hatten wir am Anfang keine Unterstützung für die Maus. Aber dank diesem „eat your own dog food“ haben wir das Werkzeug immer weiter verfeinert und verbessert.

Seit wir im Mai 2015 ein Beta von Code verfügbar gemacht haben, hilft uns jetzt auch eine sehr aktive Community, Code zu verbessern. Wir erhalten laufend Feedback über Bug-Reports, Tweets, Fragen auf Stack Overflow, und wir versuchen, darauf in unseren monatlichen Releases zu reagieren.

Tobias Kahlert und Kay Giza (von Microsoft Deutschland) gehörten zu frühen Anwendern von Code, und sie haben uns schon länger geholfen, Fragen der Community zu beantworten. Dieses E-Book ist der logische nächste Schritt. Mit einer Sammlung von Tipps und Tricks geben sie ihre Erfahrungen weiter.

Mein Ziel ist es, dass unsere Anwender Visual Studio Code genauso effektiv nutzen können, wie wir das als Entwicklungsteam von Code täglich tun. In diesem E-Book beschreiben Tobias und Kay dutzende von Tipps und Tricks, die Ihnen helfen, diesem Ziel etwas näher zu kommen.

Happy Coding!

Dr. Erich Gamma
Microsoft Distinguished Engineer
Open Tools Group, Development Lead Visual Studio Code
Zürich, März 2016

Über Erich Gamma
Erich Gamma ist ein Microsoft Distinguished Engineer und leitet das Projekt Visual Studio Code. Sein Team und er arbeiten von einem kleinen Entwicklungszentrum in Zürich aus. Das Team ist außerdem für den Online-Editor Monaco verantwortlich, der in vielen Microsoft-Produkten Einsatz findet. Zuvor war Erich für IBM tätig, wo er als einer der Väter von Eclipse für deren Java-Entwicklungswerkzeuge verantwortlich war. Erich ist überdies Mitglied der „Gang of Four“, die durch den Buch-Klassiker „Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software“ bekannt wurde. Gemeinsam mit Kent Beck hat Erich Gamma an der Entwicklung von JUnit gearbeitet, dem de-facto Standard-Testwerkzeug für Java-Software.

Was ist Visual Studio Code?

Visual Studio Code (VS Code) ist ein Open Source **Code Editor** zum Entwickeln und Debuggen moderner Cloud- und Webanwendungen, der kostenlos für Linux, OS X sowie Windows verfügbar ist. VS Code unterstützt mehr als 30 unterschiedliche Programmier-, Auszeichnungs- und Datenbanksprachen, von denen als Beispiele hier nur JavaScript, C#, C++, PHP, Java, HTML, R, CSS, SQL, Markdown, TypeScript, Less, Sass, JSON, XML und Python genannt werden sollen.

Der blitzschnelle Editor kann dabei nicht nur mit integriertem Debugging (auch für ASP.NET 5- und Node.js-Anwendungen) überzeugen: Sogar Git-Unterstützung ist auf Knopfdruck verfügbar. Zahlreiche Tastenkombinationen sorgen für Produktivität bei der Arbeit.

Anders als von Visual Studio 2015 und Vorgängern gewohnt arbeitet VS Code nicht mit Projektdateien, sondern auf Datei- und Ordner-Ebene. Die Umgebung kann durch Extensions flexibel erweitert werden, unabhängig vom verwendeten Betriebssystem. Code erweist sich so als **die perfekte Ergänzung der bevorzugten Entwicklungstools**.

Visual Studio Code bleibt dank integrierter Update-Funktion stets auf der Höhe der Zeit. Monatliche Updates sorgen für eine dynamische Fortentwicklung, die für alle unterstützten Plattformen zeitgleich vorangetrieben wird. Und mit Visual Studio Code „Insiders“ steht sogar eine parallel installierbare Option zur Verfügung, die stets die neuesten Features als Vorschauversion bietet.

Einige Hinweise zum E-Book

Dieses E-Book ist im März 2016 auf Basis von englischsprachigen Vorabversionen entstanden. Visual Studio Code entwickelt sich tagtäglich weiter, und neue Versionen werden in monatlichen Zyklen veröffentlicht. Selbst ein E-Book kann mit solchen Meilenstiefeln nur schwer Schritt halten. Basierend auf Ihrem Feedback erwägen wir daher, dieses Buch auf GitHub zu transferieren, so dass alle Interessierten an den Tipps und Tricks weiterarbeiten können. Sagen Sie uns doch im deutschsprachigen Forum zu Visual Studio Code Ihre Meinung dazu.

Außerdem möchten wir Sie als Mac- oder Linux-Anwender darauf hinweisen, dass unser Buch auf Windows-Basis entstanden ist und dessen Tastenbefehle nutzt. Sie finden alle Tastenzuordnungen für Ihr Betriebssystem auf code.visualstudio.com.

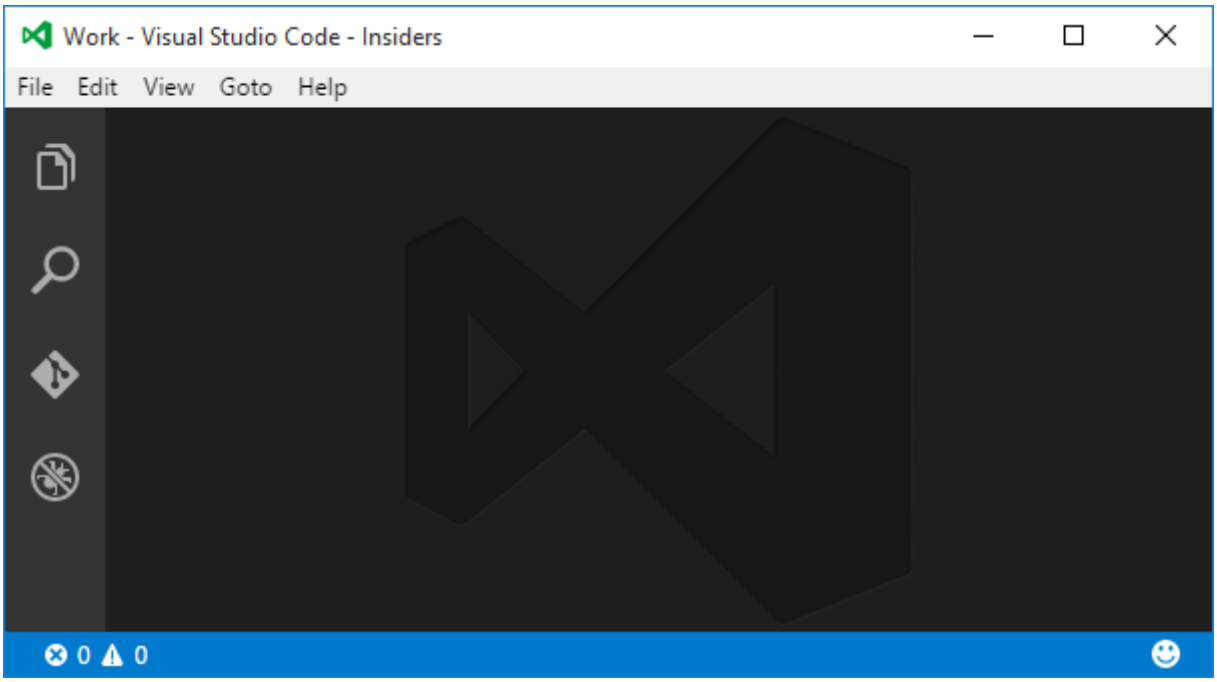
Zuletzt möchten wir Ihnen noch einen Hinweis auf die neue Tipp-Sammlung des VS Code Teams geben, denn Tipps und Tricks kann man nie genug kennen. Code-Pakete und Ressourcen für VS Code finden Sie außerdem in einem weiteren wachsendes Repository, das wir Ihnen auch ans Herz legen.

1. Tipp – Updates vor allen anderen erhalten als VS Code Insider

Visual Studio ist Open Source und wird ständig weiterentwickelt. Auch wenn Änderungen sofort im öffentlichen Repository unter https://github.com/Microsoft/vscode zu sehen sind, dauert es doch eine Zeit lang, bis diese es in die veröffentlichten Releases von VS Code schaffen.

Wer Vorabversionen von VS Code testen möchte, kann sich zusätzlich zur normalen Version auch noch die Insiders-Version installieren. Diese ist, neben der normalen Version von VS Code, ein eigenständiges Programm mit eigenen Einstellungen und Erweiterungen, das problemlos parallel installiert werden kann. So kann man die Insiders-Version benutzen, um die neusten Features zu testen – und sollte diese an einem bestimmten Punkt nicht wie erhofft funktionieren, kann man einfach zur normalen Version wechseln.

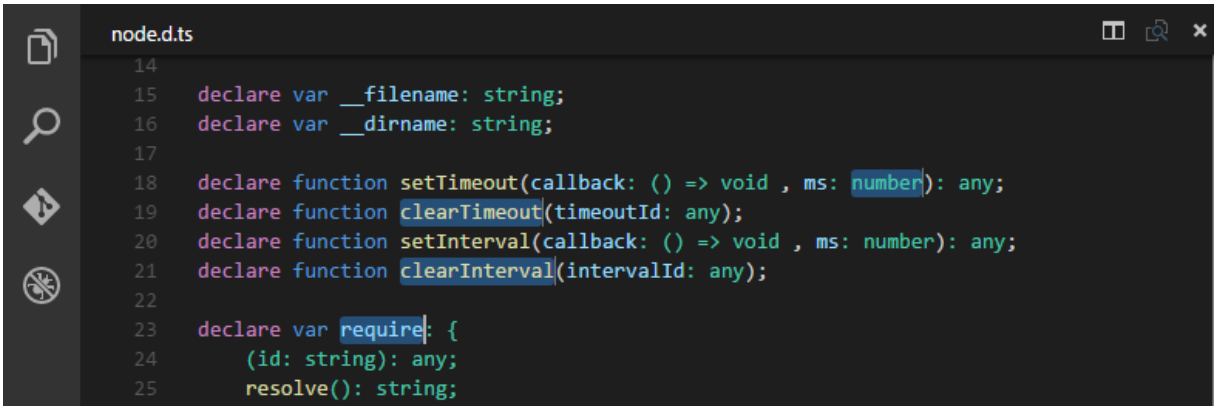
Die Insiders-Version kann unter https://code.visualstudio.com/insiders heruntergeladen werden.
#VisualStudioCodeInsiders



2. Tipp – Mehrere Eingabemarken

Visual Studio bietet die Möglichkeit, mehrere Stellen in einer Datei parallel zu bearbeiten. Dies kann beispielsweise nützlich sein, wenn viele identische Änderungen vorgenommen werden sollen.

Um eine zusätzliche Eingabemarke an einer bestimmten Stelle zu erzeugen, muss im Texteditor mit gedrückter **Alt**-Taste auf diese Stelle geklickt werden. Jede Eingabe auf der Tastatur wird nun an jeder definierten Eingabemarke durchgeführt. Es ist außerdem möglich, verschiedene Stellen im Code gleichzeitig zu markieren.



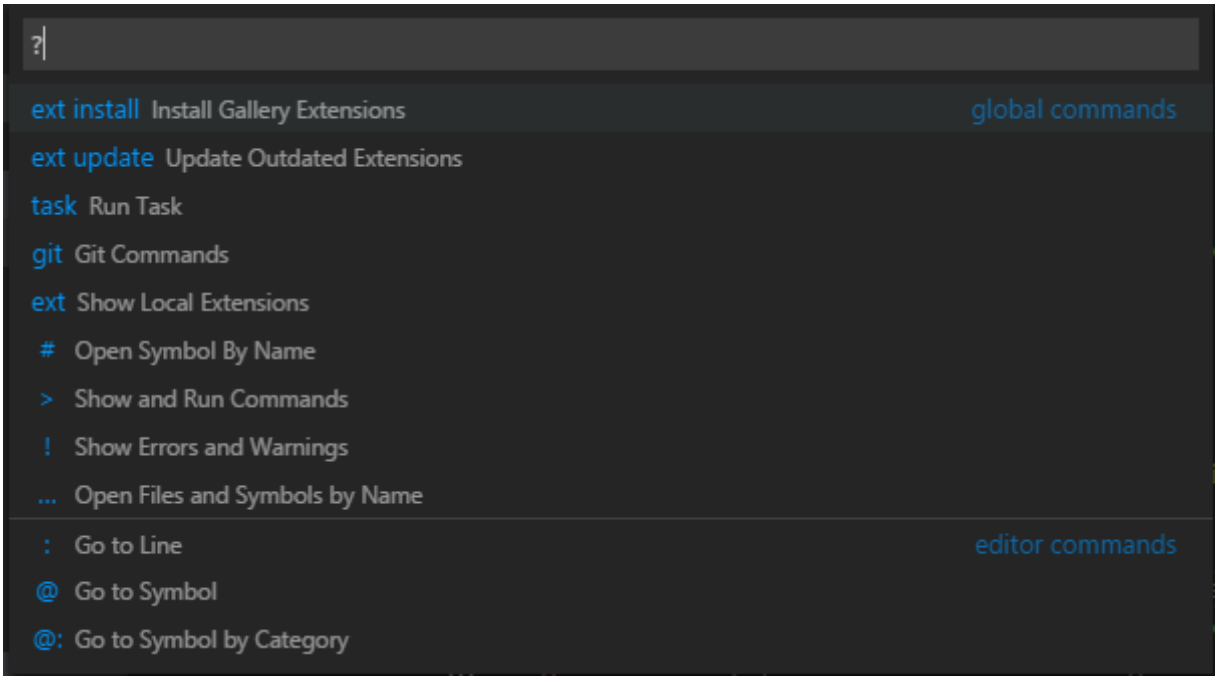
3. Tipp – Visual Studio Code mit der Kommandopalette steuern

Beim Benutzen von Visual Studio Code kommt man zwangsläufig mit der Kommandopalette in Berührung. Meist wird sie mit **F1** geöffnet und ermöglicht dann die Eingabe von Kommandos.

Allerdings kann sie noch viel mehr, denn bei genauerem Hinsehen erkennt man, das nach Druck auf **F1** bereits ein **„>“** in der Eingabezeile steht. Das erste Zeichen oder Wort bestimmt den Modus der Eingabe. Das **„>“**zum Beispiel aktiviert den Kommando-Modus.

Um von vornherein eine leere Eingabe zu öffnen, steht die Tastenkombination **Strg+P** zur Verfügung. Damit wird der Navigations-Modus aktiviert: In einer Liste werden zunächst die zuletzt geöffneten Dateien angezeigt. Nach Eingabe einiger Buchstaben werden die geöffneten Dateien (und die Dateien im geöffneten Ordner) gefiltert und angezeigt. So lässt sich sehr schnell zwischen Dateien wechseln. Befindet man sich in einem Projekt, welches IntelliSense unterstützt, werden sogar die gefundenen Symbole berücksichtigt.

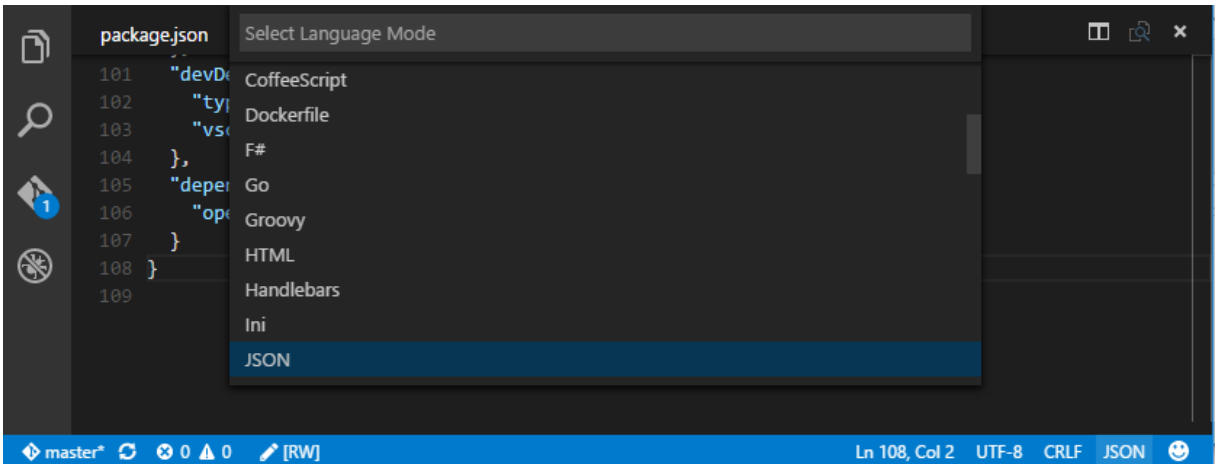
Es gibt noch weitere Modi, mit denen sich viele Bereiche von VS Code steuern lassen. Eine Liste aller Kombinationen wird nach Eingabe von **„?“** angezeigt. Es lohnt sich wirklich, sich die verschiedenen Möglichkeiten einzuprägen und sie einzusetzen: Mit ihrer Hilfe kann man sich oft den Griff zur Maus sparen und sein Arbeitstempo erhöhen.



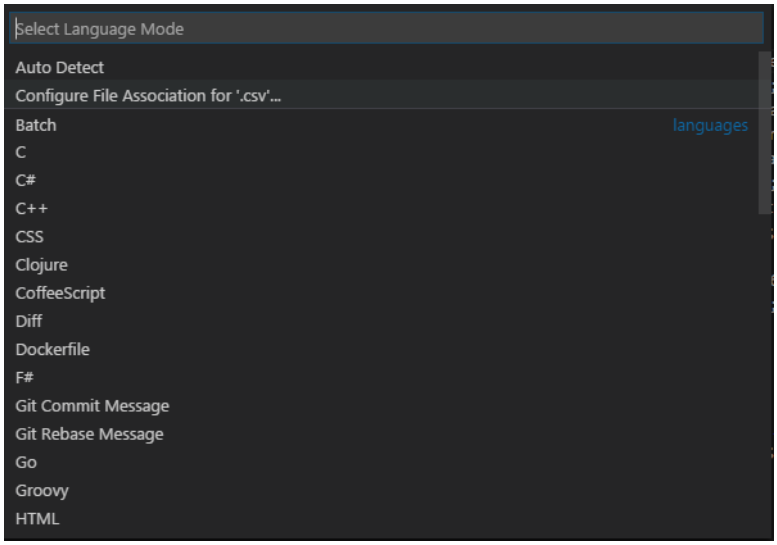
4. Tipp – Sprache für eine Datei auswählen

Ab und zu kommt es vor, dass VS Code die in der geöffneten Datei verwendete Sprache nicht erkennt. Das ist häufig dann der Fall, wenn man eine neue Datei erzeugt oder eine Datei keine Dateiendung aufweist. VS Code unterstützt Syntax-Highlighting (Syntaxhervorhebung) für sehr viele Programmiersprachen, also wäre es sehr bedauerlich, davon nicht zu profitieren.

Um im Problemfall die Sprache für die aktive Datei auszuwählen, kann entweder das Kommando **Change Language Mode** ausgeführt werden oder unten rechts in der Statusleiste auf die aktuell ausgewählte Sprache geklickt werden. Es öffnet sich dann eine Liste mit unterstützten Sprachen, aus der man die gewünschte auswählen kann.



Auch in einem weiteren Fall kann VS Code eine Sprache möglicherweise nicht erkennen: Dann nämlich, wenn einem Dateinamens-Suffix keine Sprache zugeordnet ist. Im Auswahlfenster für Sprachen kann man diese Verknüpfung in einem solchen Fall dann vornehmen.



5. Tipp – AutoSave

VS Code unterstützt AutoSave. Damit wird das momentan geöffnete Dokument nach jeder Änderung automatisch gespeichert. Gerade für Dateien im Web-Bereich ist das ungemein praktisch: So muss nicht ständig auf Speichern geklickt werden, und die Dateien sind immer aktuell, wenn man im Browser auf „Aktualisieren“ klickt.

Vorsichtig sollte man allerdings sein, wenn man automatische Tasks benutzt, die immer dann neu anlaufen, wenn sich eine Datei ändert (zum Beispiel, um TypeScript zu JavaScript zu transpilieren). Das kann unter anderem die CPU sehr stark auslasten.

Die AutoSave-Funktion lässt sich wie üblich in den Einstellungen mit der Option **"files.autoSave"** aktivieren oder deaktivieren. Außerdem ist es möglich, die AutoSave-Funktion erst nach einem definierten Zeitintervall auszuführen: Dafür wird **"files.autoSave"** auf **"afterDelay"** gesetzt und die Eigenschaft **"files.autoSave"** auf das Verzögerungsintervall in Millisekunden, z. B.:

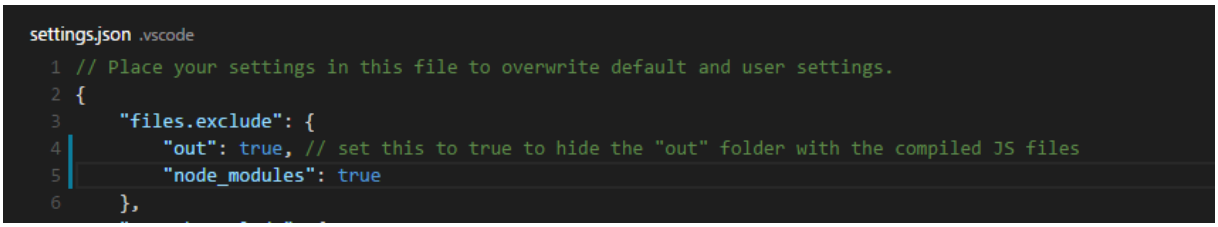
```
"files.autoSave": "afterDelay"
"files.autoSaveDelay": 1000
```

6. Tipp – Unwichtige Ordner verstecken

Hat man mit VS Code einen Ordner geöffnet, in welchem sich Dateien oder ganze Ordner befinden, die man niemals editiert, so können diese einfach aus der Explorer-Ansicht ausgeblendet werden. In einem Node.js-Projekt könnte das zum Beispiel der Ordner **node_modules** sein.

Zum Ausblenden muss der Ordner, die Datei oder ein beliebiges Pattern in die globalen Einstellungen oder in den Workspace-Einstellungen in die Option **"files.exclude"** eingetragen werden. Die Workspace-Einstellungen bieten sich hier natürlich an, da die angegebenen Dateien dann wirklich nur in diesem Projekt ausgeblendet werden.

Um die Workspace-Einstellungen zu öffnen, reicht es aus, den Befehl **Preferences: Open Workspace Settings** auszuführen. Existiert zu diesem Zeitpunkt noch keine Datei mit den Workspace-Einstellungen, so wird VS Code automatisch eine neue Datei dafür erzeugen.



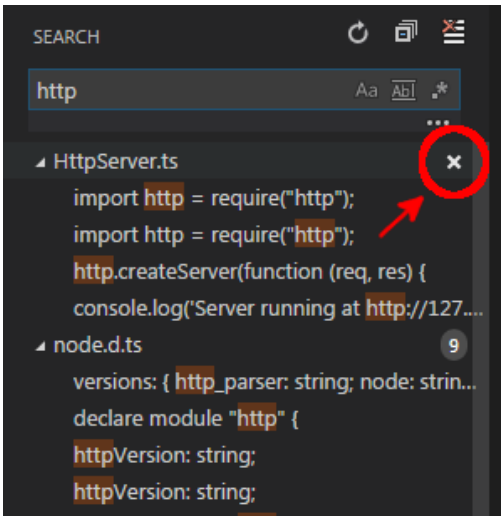
Ein nützliches Beispiel für ein Pattern dient dem Ausblenden aus TypeScript generierter JavaScript-Ressourcen:

```
"files.exclude": {
        "**/*.js": { "when": "$(basename).ts"}
}
```

7. Tipp – Suchergebnisse ausblenden

Visual Studio Code bietet eine wunderbare Suche über allen geöffneten Dateien hinweg. Beim Durchgehen aller Suchergebnisse kann es hilfreich sein, bereits untersuchte Treffer aus der Liste auszublenden: Muss an den gefundenen Stellen etwas geändert werden, so hilft dies, den Überblick über die bereits bearbeiteten Stellen zu behalten.

VS Code zeigt neben allen Dateien, in denen die Suche etwas gefunden hat, zusätzlich an, wie viele Fundstellen es in der jeweiligen Datei gab. Wird der Mauszeiger über diese Zahl bewegt, so erscheint ein Kreuz. Ein Klick darauf sorgt dafür, dass die zugehörige Datei aus der Liste ausblendet wird.



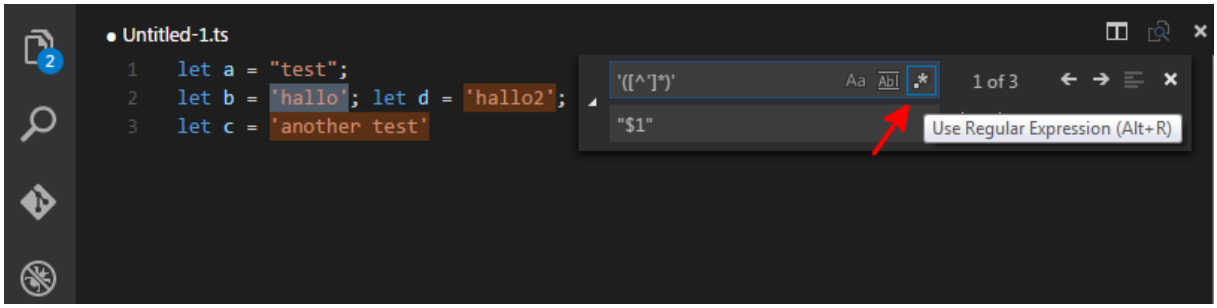
Nach einem erneuten Ausführen der Suche werden alle Dateien wieder aufgelistet.

8. Tipp – RegEx-Treffer beim Ersetzen wieder einsetzen

Visual Studio Code bietet eine Suche, die auch reguläre Ausdrücke unterstützt. So lassen sich nicht nur Texte finden, sondern ganze syntaktische Konstrukte.

Besonders mächtig wird diese Funktion, wenn man sie mit der Ersetzen-Funktion kombiniert. Diese kann nicht nur statischen Text einsetzen, sondern sich auch auf die Ergebnisse der regulären Suche beziehen. Mit Klammern können dafür in regulären Ausdruck Gruppen definiert werden. Die Texte, die bei der Suche dann für diese Gruppen gefunden werden, können über **$1**, **$2**, **$3**,… im Ersetzungs-Text referenziert werden.

Möchte man zum Beispiel einheitlich alle Texte mit umgebenden Hochkommata ermitteln und letztere durch Anführungszeichen ersetzen, so gibt man in der Suchzeile **'([^']*)'** und in der Ersetzungszeile **"$1"** ein: Der erste Ausdruck findet alle Strings mit Hochkommata, und der von ihnen umschlossene Text wird durch die Klammern als Gruppe 1 definiert. Der zweite Ausdruck ersetzt den kompletten gefundenen Text mit zwei Anführungszeichen und dem Inhalt des ursprünglichen Texts (nämlich dem Wert der Gruppe 1).



Soll das Dollarzeichen selber als Term verwendet werden, so erfolgt dies durch Verdoppelung des Zeichens ($$).

9. Tipp – Emmet Snippets

HTML und XML sind trotz der leichten Lesbarkeit oft mühsam zu schreiben. Um das Schreiben solcher Dokumente zu beschleunigen, unterstützt VS Code *Emmet Snippets*. Emmet Snippets sind CSS-ähnliche Ausdrücke, die mit Betätigung der Tab-Taste beispielsweise in HTML oder XML umgewandelt werden.

Im einfachsten Fall wird nur ein Element erzeugt. Schreibt man zum Beispiel **„html“** in eine Datei, die VS Code als HTML-Datei erkennt, und drückt dann daraufhin die Tab-Taste, wird dieser Text durch **„<html></html>“** ersetzt. Die Eingabemarke wird dabei zwischen den Tags platziert.

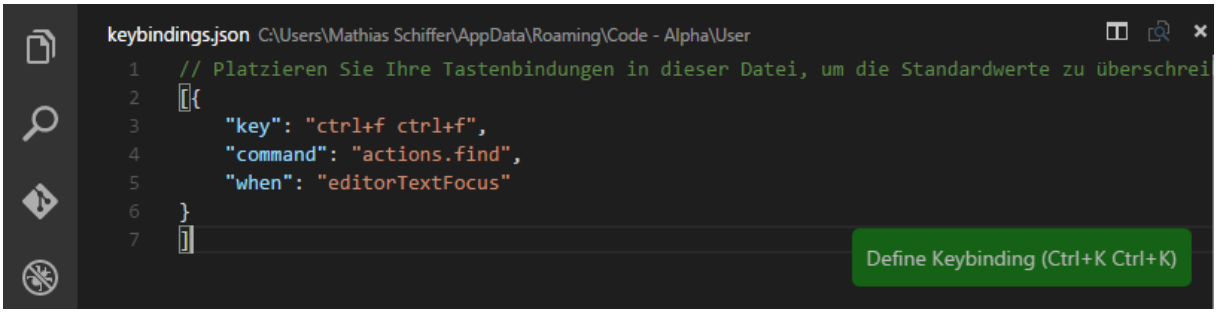
Doch Emmet Snippets können noch mehr. So gibt es eine ganze Reihe von Konstrukten, die von Emmet Snippets unterstützt werden: Drückt man etwa nach der Eingabe von „li*5“ die Tab-Taste, wird 5 Mal der Text **„
“** eingefügt. Unter http://docs.emmet.io/cheat-sheet/ sind verschiedenste solcher Emmet-Konstrukte zu finden.

Neben HTML und XML unterstützt VS Code auch weitere Formate: Derzeit sind diese Razor, CSS, Less, SCSS, XSL, Jade, Handlebars, HBS, JSX und TSX.

10. Tipp – Tastenkürzel-Akkorde

Wie schon Visual Studio, unterstützt auch Visual Studio Code Tastenkürzel-Akkorde. Das bedeutet, dass der Editor nach Eingabe einer Tastenkombination auf eine zweite Tastenkombination wartet. Auf diese Weise lassen sich ähnliche Befehle gruppieren.

Zum Beispiel kann man die lokale Suche in einer Datei von der ursprünglichen Kombination **Strg+F** auf den Akkord **Strg+F Strg+F** legen und die globale Suche in allen Dateien auf **Strg+F Strg+G**. In der **keybindings.json** müssen die beiden Bestandteile eines Akkords durch Leerzeichen getrennt werden.

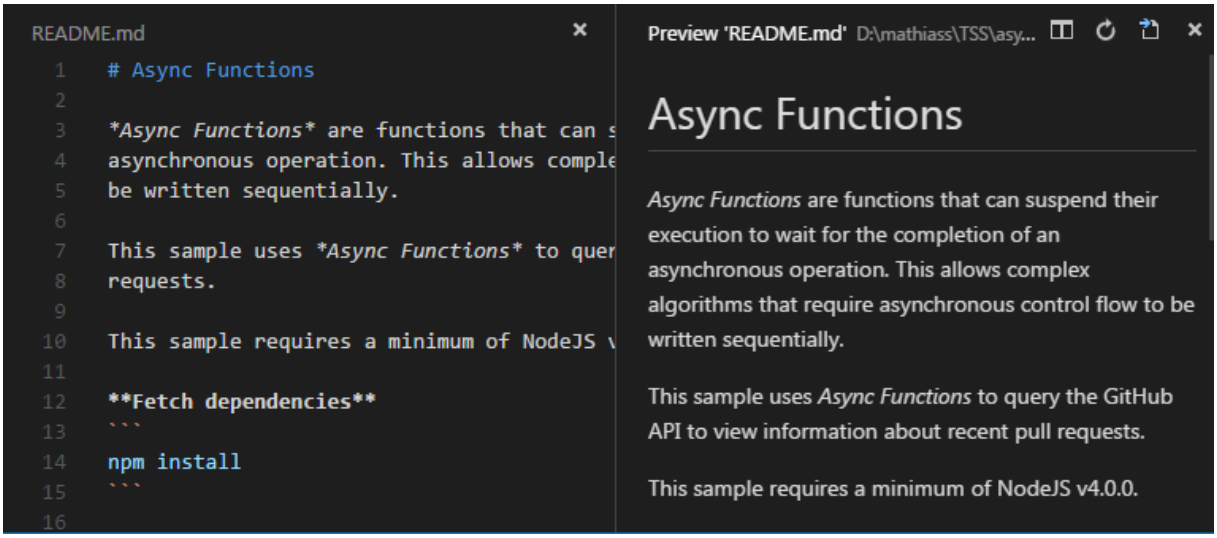


Nach dem Druck des ersten Tastenkürzels wartet Visual Studio Code auf weitere Eingaben. Das wird auch unten in der Statusbar angezeigt: **„(Ctrl+F) was pressed. Waiting for second key of chord…“**.

11. Tipp – Markdown-Vorschau

Markdown ist eine einfache Auszeichnungssprache, um formatierte Texte zu verfassen. Auch wenn Markdown in seiner Ausgangsform bereits gut lesbar ist, ist es noch angenehmer zu lesen, wenn es als ein richtig formatiertes Dokument angezeigt wird – gerade was externe Ressourcen wie Bilder angeht.

Visual Studio Code unterstützt eine Markdown-Vorschau, die neben dem Text geöffnet werden kann. VS Code benutzt hier das GitHub Flavored Markdown, welches zusätzliche Funktionen zum normalen Markdown bietet. Das ist besonders bei GitHub-Readmes hilfreich.



Mithilfe von **Strg+Umschalttaste+V** kann dabei zwischen Markdown und Vorschau hin- und hergewechselt werden. Der Akkord **Strg+K, V** öffnet die Vorschau auf der Seite. In diesem Kontext ist auch die Umschaltung von Zeilenumbrüchen (**Alt+Z**) besonders erwähnenswert, die auch über den Punkt **Toggle Word Wrap** des **View**-Menüs gesteuert werden kann.

12. Tipp – CSS-Selektoren visualisieren

CSS-Selektoren stehen am Anfang jeder CSS-Klausel und bestimmen, auf welche HTML-Elemente die nachfolgenden Eigenschaften zutreffen sollen. Diese Selektoren können mitunter recht kompliziert werden – unter anderem, weil ab und zu doch ein Konstrukt dabei ist, welches man nicht so häufig benutzt.

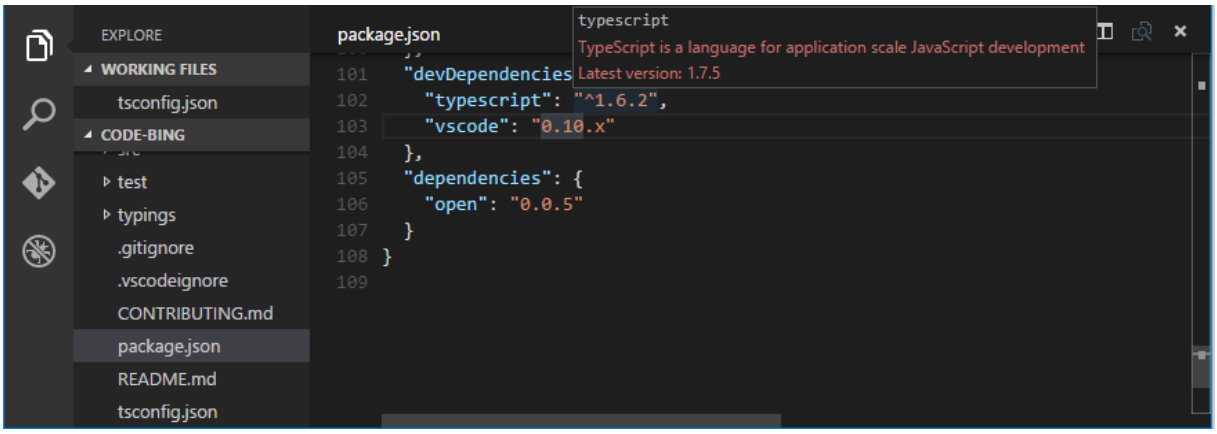
Visual Studio Code hilft hier weiter, indem es in einer Vorschau für den unter dem Mauszeiger befindlichen Selektor anzeigt, wie die HTML-Elemente später aussehen müssen, damit der Selektor angewendet wird.

13. Tipp – „Latest Version“ von Abhängigkeiten in package.json

Arbeitet man an einem Node.js-Projekt, gibt es in der **package.json** mehrere Abschnitte, die die Abhängigkeiten definieren. Zu jeder Abhängigkeit müssen auch eine oder mehrere gewünschte Versionen angeben werden.

VS Code bietet hier die Möglichkeit, Informationen zu dem Modul anzuzeigen. Dazu muss der Mauszeiger einfach über die Abhängigkeitsangabe bewegt werden. Angezeigt werden dann der Name, eine kurze Beschreibung und die neuste Version des Moduls.

Über **Strg+Leertaste** kann überdies eine neue Abhängigkeit auf ein neues Modul erfasst werden.

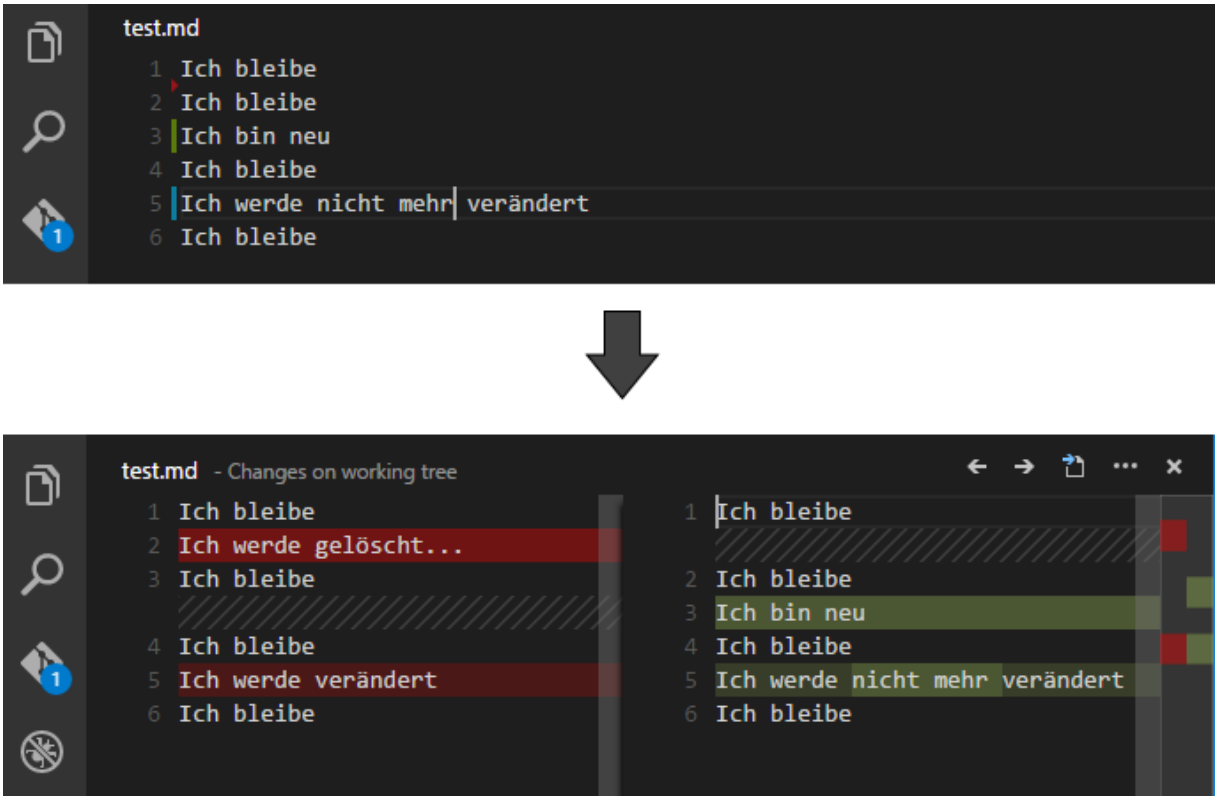


14. Tipp – Git Quick Change Info

Arbeitet man in einem Git-Repository, bietet VS Code über die Git-Vergleichsansicht eine wunderbare Möglichkeit, die Arbeitsversion mit der aktuellen Repository-Version zu vergleichen.

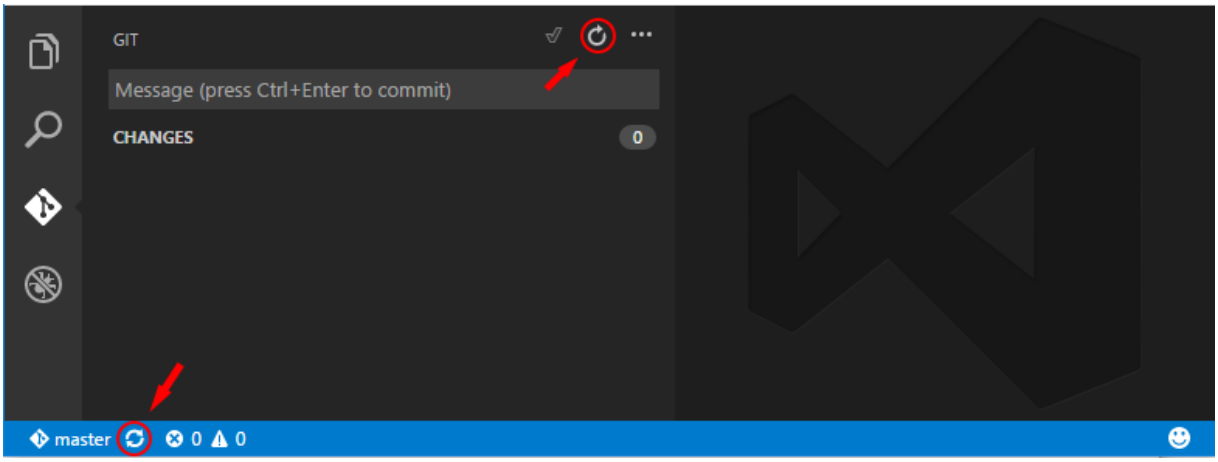
Aber auch schon beim Editieren einer Datei, die unter Git Source-Verwaltung steht, zeigt VS Code Änderungen zur vorhergehenden Version an. An der linken Seite des Editors (zwischen Zeilennummern und Quelltext) werden geänderte, gelöschte und neue Zeilen markiert.

- Grüne Linie: Zeilen mit grüner Linie sind hinzugekommen
- Blaue Linie: Zeilen mit blauer Linie wurden verändert
- Roter Pfeil: Bei einem roten Pfeil wurden Zeilen gelöscht



15. Tipp – Git-Ansicht aktualisieren

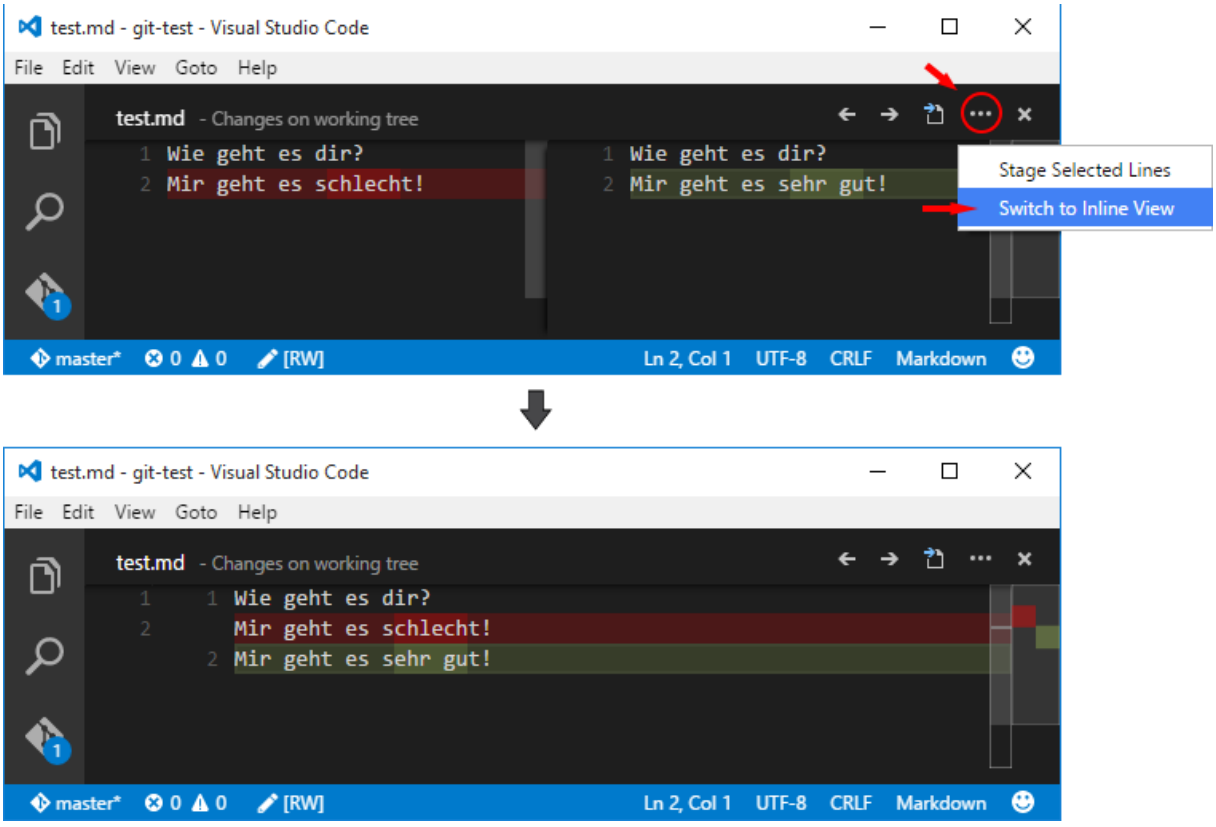
Visual Studio Code benutzt das externe Programm Git, um die Änderungen in einem öffentlichen Git-Verzeichnis zu finden. Dabei kann es schon einmal vorkommen, dass die Ansicht nicht ganz aktuell ist oder durcheinandergerät. Um die Git-Ansicht wieder auf den Stand der Dinge zu bringen, gibt es oben in der Git-Ansicht und unten in der Statusbar Refresh-Knöpfe.



16. Tipp – Git-Inline-Vergleich

Lässt man sich von Visual Studio Code die Änderungen einer Datei zur aktuellen Git-Version zeigen, so öffnet sich eine Ansicht, in der die Änderungen nebeneinander gezeigt werden.

Visual Studio Code bietet aber auch die Möglichkeit, die Änderungen in einer Ansicht mit nur einem Code-Fenster zu zeigen: Dazu gibt es im Reiter ein kleines Untermenü mit dem Menüpunkt **Switch to Inline View**. Ein Klick darauf wechselt zum Git-Inline-Vergleich.



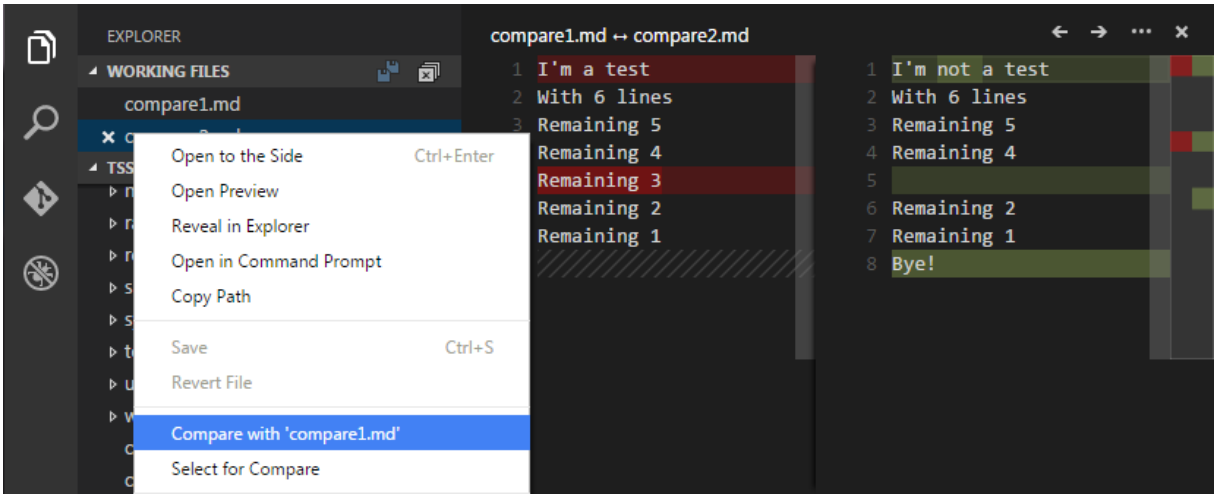
Um zur **Side by Side View** zurück zu wechseln, benutzt man einfach dasselbe Menü nochmals.

17. Tipp – Dateien vergleichen

VS Code bietet eine großartige Ansicht, um zwei Dateien zu vergleichen. Insbesondere beim Benutzen von Git mit VS Code wird man oft auf diese Ansicht stoßen. Klickt man im Git-Bereich auf eine Datei, werden die Änderungen zur vorherigen Version der Datei angezeigt.

VS Code kann aber nicht nur Versionen einer Datei vergleichen, sondern auch jede beliebige Datei mit einer anderen. Die Vergleichsansicht kann entweder über die Kommandopalette aufgerufen werden oder über das Kontextmenü im Explorer-Bereich.

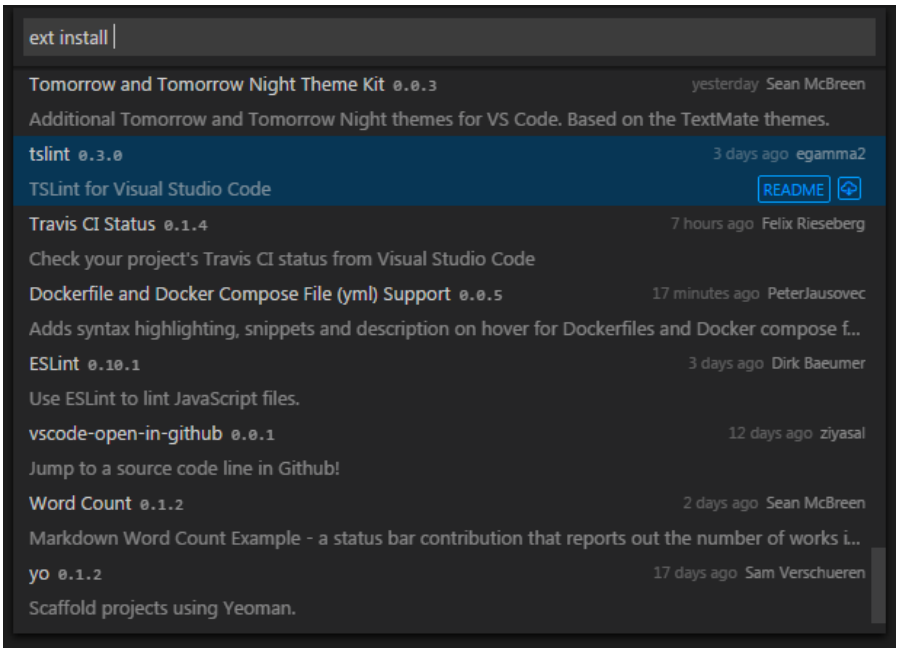
Im ersten Fall ruft man das Kommando **Files: Compare Active File With...** auf und wählt dann aus einem Menü eine Datei, mit der man die momentan geöffnete Datei vergleichen will. Im zweiten Fall öffnet man mit der rechten Maustaste das Kontextmenü einer Datei in der Explorer Ansicht und wählt **Select for Compare**. Öffnet man ein weiteres Kontextmenü einer anderen Datei, kann man nun **Compare with ‚<Erste Datei>'** auswählen. In beiden Fällen öffnet sich daraufhin eine Vergleichsansicht.



18. Tipp – Extensions

Visual Studio besitzt von Haus aus bereits einen ganzen Haufen an Funktionen. Das Beste von allen Features ist aber, neue Funktionalität zu entwickeln und auf einfache Weise mit dem Rest der Welt zu teilen: Extensions. Das sind Erweiterungen, die mit TypeScript oder JavaScript entwickelt und über ein spezielles Programm im Visual Studio Code Marketplace veröffentlicht werden. Andere Nutzer können dann diesen Marketplace durchsuchen und die zu Verfügung stehenden Extensions mit wenigen Klicks installieren. So lässt sich VS Code von dritter Seite erweitern.

Alle Extensions können unter https://marketplace.visualstudio.com/#VSCode eingesehen werden und dann mit dem Kommando **Extension: Install Extension** installiert werden.

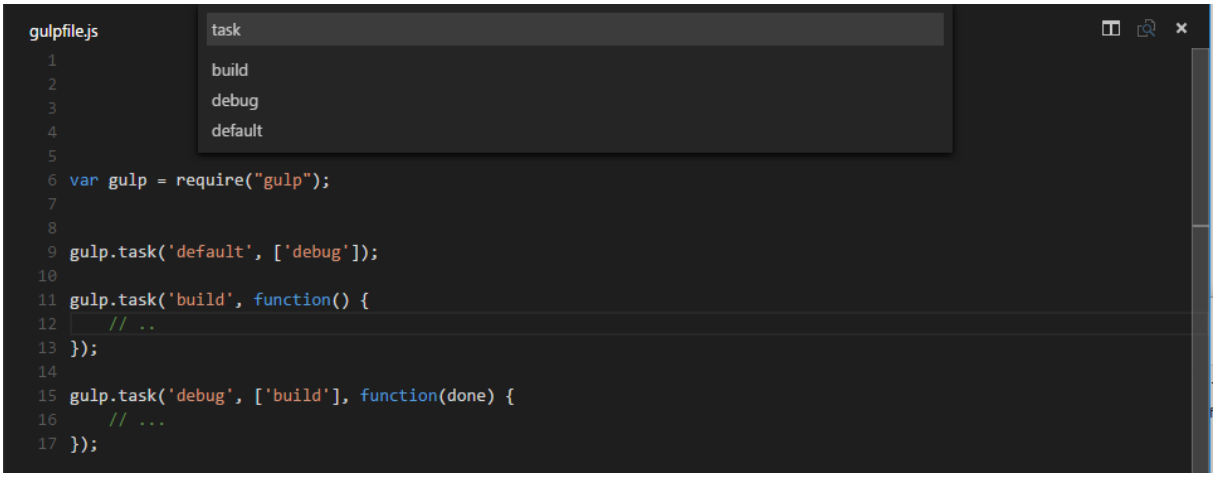


Weitere Informationen zu Extensions gibt es unter Visual Studio Code: Jetzt Open Source und erweiterbar!. Das Tollste an Extensions ist natürlich, dass man sie auch selber entwickeln kann. Dazu gibt es unter Visual Studio Code: Extensions entwickeln eine Anleitung mit einem kleinen Beispiel.

19. Tipp – Automatische Taskerkennung für Gulp, Grunt und Jake

Mit Tasks können in Visual Studio Code allerlei externe Programme gestartet werden. Dazu trägt man die entsprechenden Parameter und Optionen in eine **.vscode/tasks.json** ein und kann die Tasks dann über die Kommandopalette starten.

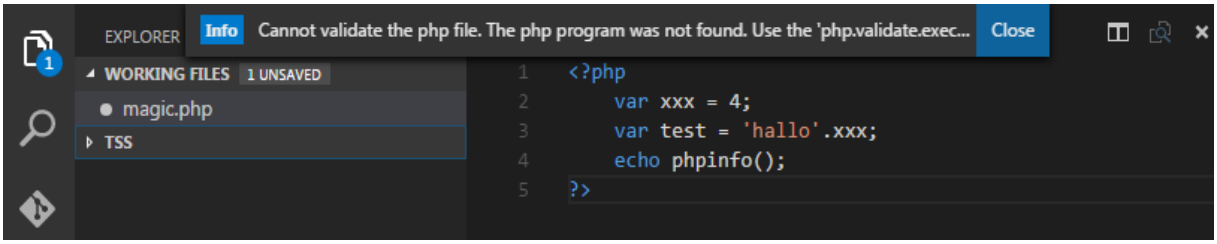
Arbeitet man aber an einem Node.js Projekt und benutzt Gulp, Grunt oder Jake, kann VS Code die dort definierten Tasks automatisch erkennen. Das Schreiben einer **tasks.json** entfällt somit. Man führt dann einfach das Kommando **Tasks: Run Task** aus, und es öffnet sich eine Liste mit Tasks, aus denen man auswählen kann.



20. Tipp – PHP-Konfiguration in Visual Studio Code

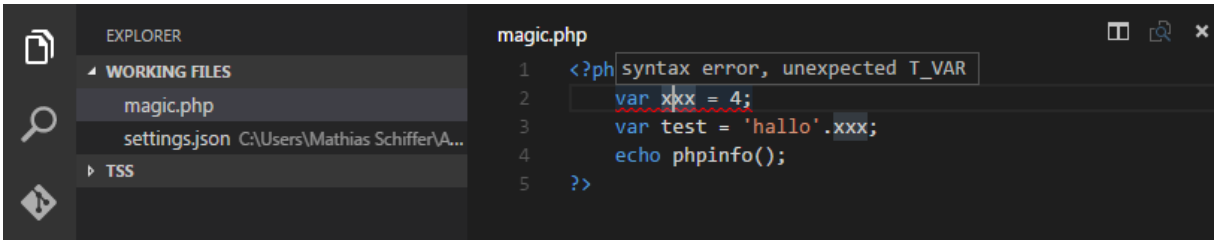
Visual Studio Code unterstützt von Haus aus Syntax-Highlighting für PHP. Auch Hilfstexte und Parameterinfos stehen für viele Funktionen bereit.

VS Code kann aber auch den Quelltext überprüfen und Fehler live anzeigen. Dazu muss allerdings der Pfad zu einer PHP-Version eingestellt werden. Darauf wird man auch hingewiesen, wenn man eine beliebige PHP-Datei öffnet. Es kommt dann ggf. die folgende Benachrichtigung: **“Cannot validate the php file. The php program was not found. Use the 'php.validate.executablePath' setting to configure the location of 'php'”**.



Hat man noch kein PHP auf dem Rechner, kann man es sich von http://php.net/downloads.php herunterladen und lokal installieren (bzw. das Archiv an einen Ort entpacken). In dem PHP-Ordner befindet sich dann eine Datei namens php.exe (unter Windows). Der Pfad zu dieser Datei muss dann in der Visual Studio Code Einstellung **"php.validate.executablePath"** eingetragen werden.

Beim nächsten Öffnen einer PHP-Datei erhält man nun die Fehlerstellen angezeigt.



Tipp: Um mit Visual Studio Code auch PHP debuggen zu können, gibt es im Marketplace eine Extension: PHP Debug adapter for Visual Studio Code.

21. Tipp – Kommandozeilenparameter von Visual Studio Code

Visual Studio Code wird bei der Installation der PATH-Umgebungsvariable hinzugefügt und lässt sich über den Befehl **code** aus der Konsole heraus starten. Das ist hilfreich, wenn Code zum Beispiel von anderen Programmen geöffnet werden soll. Mit der Eingabe von **code** öffnet sich aber lediglich ein leerer Editor. Welche Parameter unterstützt werden, wird im Folgenden beschrieben:

Zunächst mal sieht VS Code alle Parameter als Pfade zu Dateien oder Ordnern. Diese werden dann in einem neuen VS Code-Fenster geöffnet. Werden mehrere Ordner angegeben, öffnen sich auch mehrere Fenster. Vor, zwischen und nach den Pfad-Parametern können eine Reihe von speziellen Optionen angegeben werden. Die gebräuchlichsten davon sind nachfolgend aufgelistet:

-r, --reuse-window	Öffnet angegebene Dateien in einem bestehenden Fenster von VS Code oder öffnet ein neues.
-n, --new-window	Zwingt VS Code, in jedem Fall ein neues Fenster zu öffnen.
-g, --goto	Pfade im Format **„Dateipfad:Zeile:Spalte“** werden mit der Eingabemarke an der angegebenen Stelle geöffnet.
--disable-extensions	Öffnet VS Code, ohne installierte Extensions zu laden.
--verbose	Lässt VS Code Debug-Informationen ausgeben
--diff <file1> <file2>	Öffnet den Diff-Editor mit den beiden angegebenen Dateien.

Konsolenunterstützung unter Linux

Um Visual Studio Code auch unter Linux aus der Konsole heraus starten zu können, muss ein symbolischer Link von /usr/local/bin/code zur ausführbaren Datei von VS Code erzeugt werden. Das kann mit folgendem Befehl bewerkstelligt werden:

```
sudo ln -s /path/to/vscode/Code /usr/local/bin/code
```

22. Tipp – Speicherorte in Visual Studio Code

Visual Studio speichert verschiedene Dateien an unterschiedlichen Orten im Dateisystem ab. Mitunter ist es sehr hilfreich zu wissen, was wo gespeichert wird.

Einstellungen, Tastenkombinationen und Code-Snippets

Einstellungen sind in VS Code leicht über die Kommando-Palette zu erreichen. Im Dateisystem werden sie sich in der Datei **settings.json** abgebildet. Je nach Betriebssystem ist diese Datei an einer anderen Stelle zu finden. Neben der **settings.json** findet sich hier auch die **keybindings.json**, in der alle Tastenkombinationen gespeichert sind, und ein Ordner mit dem Namen *snippets*, in dem die Snippets für den Benutzer für verschiedene Sprachen abgelegt sind.

- **Windows**: **%APPDATA%\Code\User**
- **Mac**: **$HOME/Library/Application Support/Code/User/**
- **Linux:** **$HOME/.config/Code/User/**

Nutzer der Insider-Version finden die entsprechenden Pfade übrigens unter **%APPDATA%\Code Insiders** bzw. **.vscode-insiders**.

Extensions

Auch Extensions werden von VS Code in einen bestimmten Ordner installiert. Jede Extension bekommt dort einen neuen Unterordner, dessen Name sich aus dem Namen der Extension und dem Autor zusammensetzt. Visual Studio durchsucht beim Start diesen Ordner und lädt alle dort aufgefundenen Extensions. Tatsächlich gibt es keine Datei, die eine Liste von installierten Extensions enthält. Somit reicht es zur Installation aus, den Ordner einer Extension in den folgenden Extensions-Ordner von VS Code zu verschieben:

- **Windows**: **%USERPROFILE%\.vscode\extensions**
- **Mac**: **$HOME/.vscode/extensions**
- **Linux:** **$HOME/.vscode/extensions**

23. Tipp – Einstellungen und Extensions synchronisieren

Visual Studio Code bietet selbst keine Funktion, um Einstellungen oder installierte Extensions über mehrere Rechner hinweg zu synchronisieren. Da aber alle Einstellungen in Dateien gespeichert werden, reicht es aus, diese Dateien zu synchronisieren.

Dafür kann man Dienste wie OneDrive oder Dropbox verwenden. Ein einfacher Weg ist es, die beiden Ordner, die in Tipp 22 erwähnt wurden, durch Links zu neuen synchronisierten Ordnern zu ersetzen. Vorher sollten natürlich alle Einstellungen und Extensions in die neuen Ordner verschoben werden. Die Ordner-Verknüpfungen können einfach über die Konsole erstellt werden:

- **Windows**: **mklink /d <VSCode-Ordner> <Synchronisierter-Ordner>**
- **Mac & Linux**: **ln -s <Synchronisierter-Ordner> <VSCode-Ordner>**

Sehr einfach geht die Synchronisation über mehrere Rechner hinweg unter Nutzung von GitHub GIST auch mit der Extension Visual Studio Code Settings Sync aus dem Visual Studio Marketplace.

24. Tipp – Crash-Berichte ausschalten

Stürzt VS Code ab, wird ein Crash-Bericht erstellt, der dann an Microsoft gesendet wird, um den Fehler zukünftig beheben zu können. Wer diesen Crash-Bericht nicht erstellen und senden möchte, kann die Option **"telemetry.enableCrashReporter"** in seinen Einstellungen auf **false** setzen. Die Änderung bedarf zu ihrer Wirksamkeit eines Neustarts.

25. Tipp – Übermittlung von Telemetrie-Daten abschalten

VS Code sammelt Daten darüber, wie es benutzt wird, und sendet diese an Microsoft, wo sie anonymisiert dafür verwendet werden, VS Code zu verbessern. Will man keine Daten über die Benutzung an Microsoft senden, kann man dies über das Datei-Menü über den Punkt Einstellungen in den Benutzereinstellungen abschalten. In den erscheinenden Standardeinstellungen suchen Sie den Eintrag **"telemetry.enableTelemetry": true** und hinterlegen ihn in Ihrer gewünschten **settings.json** als **"telemetry.enableTelemetry": false**.

26. Tipp – Navigieren zwischen verwendeten Dateien

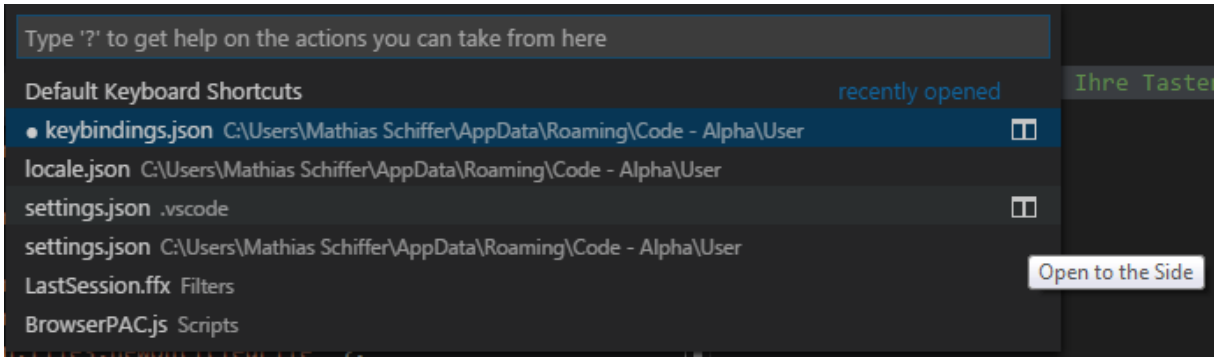
Der Editor von VS Code ist ein mächtiges Werkzeug, das seine Stärken gerade mit steuernden Tastenkombinationen ausspielen kann. Viele dieser Tastenkombinationen lassen sich bereits den Menüs von VS Code entnehmen, wo sie fein säuberlich neben den Menüpunkten aufgeführt sind, und auch die Kommandopalette (**Strg+Umschalttaste+P**) zeigt eine Vielzahl der Produktivitätshelfer an.

Besonders prominent sticht die Kombination **Strg+E** hervor, die synonym zur Tastenkombination **Strg+Tab** eingesetzt werden kann: Sie blendet eine Liste zuletzt verwendeter Dateien ein, die nach Auswahl im aktuellen Codefenster geöffnet werden können.

Die Auswahl der Datei können Sie dabei am einfachsten durch das wiederholte Betätigen der **Tab**-/**E**-Taste bei niedergehaltener **Strg**-Taste vornehmen. Das zusätzliche Betätigen der **Umschalttaste** kehrt dabei die Richtung um. Alternativ können Sie bei gedrückt gehaltener **Strg**-Taste auch die Pfeiltasten verwenden.

27. Tipp – Dateien in eigenem Editor-Fenster öffnen

Visual Studio Code unterstützt es, Dateien in einem eigenen Editor-Bereich neben dem bestehenden zu öffnen („an der Seite“). Dafür wird während des Anklickens einer Datei im Explorer-Bereich einfach zuvor die **Strg**-Taste niedergehalten. Bei Dateilisten findet sich am rechten Rand des jeweils ausgewählten Eintrags der Ausklappliste ein Icon, über das diese Funktionalität mit der Maus erreicht werden kann.



Um die Definition eines markierten Terms in einem eigenen Editorfenster zu öffnen, wird der Tasten-Akkord **Strg+K, F12** verwendet.

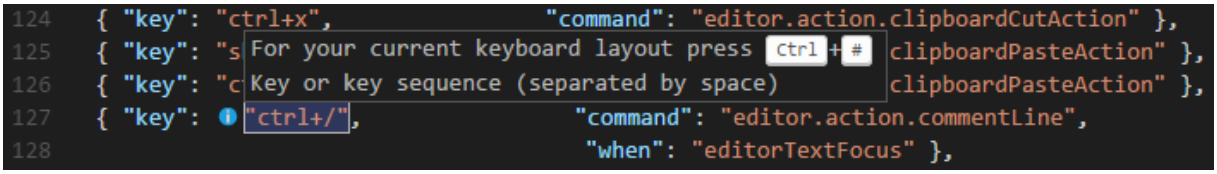
Ein direkter Wechsel zwischen den bis zu drei Editorfenstern ist über die Tastenkombinationen **Strg+1**, **Strg+2** und **Strg+3** möglich. Geschlossen werden kann jedes der Fenster über **Strg+W**.

Um den Eingabefokus auf das aktuelle Editorfenster zu legen, etwa aus dem Explorer- oder Suchfenster heraus, betätigen Sie einfach **Strg+B**. Eine erneute Betätigung kehrt zum vorherigen Zustand zurück.

28. Tipp – Tastenkombinationen anpassen

Die Bedienung der Umgebung über Tastatur ist deutlich produktiver, als die Maus zu verwenden und dafür die Hand von der Tastatur nehmen zu müssen. Auf der Website von Visual Studio Code finden Sie die jeweils aktuellen Tastenkombinationen für Ihr Betriebssystem unter [Key Bindings for Visual Studio Code](#).

Um eigene Kurzbefehle anzulegen, sollten Sie vorab die Hinweise auf der vorgenannten Webseite zum Thema internationale Tastaturen lesen: Visual Studio Code zeigt Ihnen die Tastatur-Kurzbefehle für Ihre Tastatur zutreffend an, verwendet aber intern immer das US-amerikanische Tastatur-Layout. Dies gilt auch für die Eintragungen in der Datei **keybindings.json**, die die Belegung von Tastenkombinationen definiert. Diesbezügliche Unterschiede werden Ihnen in VS Code durch einen blauen Kreis mit weißem „i“ angezeigt, der beim Überqueren mit dem Mauszeiger die zutreffenden Informationen für Ihre Tastatur einblendet.



Möchten Sie eine eigene Tastenkombination anlegen, so verwenden Sie dafür am einfachsten den Tastatur-Akkord **Strg+K**, **Strg+K**. Nachdem Sie Ihre Ziel-Tastenkombination angegeben und die Eingabetaste betätigt haben, wird in Ihrer **keybindings.json** ein vorgefertigter Eintrag für diese Kombination angelegt. Hier müssen Sie nur noch in den Werten für **command** und **when** definieren, welche Aufgabe Ihrer Tastenkombination zukommen soll. Am Ende der Standard-Tastatur-kurzbefehle, die Sie über **File** | **Preferences** | **Keyboard Shortcuts** einblenden können, finden Sie von Visual Studio Code nicht zugewiesene Befehle. Diese Liste kann in diesem Zusammenhang äußerst hilfreich sein.

29. Tipp – Lieblings-Tastenkombinationen der Autoren

Aus der großen Anzahl der verfügbaren Tastenkombinationen haben sich bei den Autoren Klassiker herauskristallisiert, die immer wieder zum Einsatz kommen, um die Arbeit zu beschleunigen. Dazu zählen (unter Windows):

- **Alt+Z** Zeilenumbruch umschalten
- **Strg+P** Schnell Dateien öffnen
- **Strg+Tab** Zwischen zuletzt verwendeten Dateien navigieren
- **Strg+1/2/3** Geöffnete Editor-Fenster anspringen
- **Strg+W** Aktiven Editor schließen
- **Strg+#** Zeile(n) ein- oder auskommentieren
- **Strg+Umschalt+K** Zeile(n) löschen
- **Strg+Klick** Datei aus dem Explorer an der Seite öffnen
- **Alt+C** Suche nach exakter Schreibweise ein/aus
- **Alt+W** Suche nach ganzen Wörtern ein/aus
- **Strg+K, F12** Definition des markierten Terms an der Seite öffnen
- **Strg+D** Auswahl auf nächsten Suchtreffer positionieren
- **Strg+K, Strg+D** Letzte Auswahl auf nächsten Suchtreffer positionieren
- **ALT+Pfeil oben** Zeile komplett nach oben verschieben
- **ALT+Pfeil unten** Zeile komplett nach unten verschieben

30. Tipp – Änderung der Oberflächensprache

Die Oberfläche von Visual Studio Code richtet sich nach den Ländereinstellungen Ihres Systems, sofern dessen Sprache unterstützt wird. Anderenfalls wird die US-amerikanische Ansicht dargestellt.

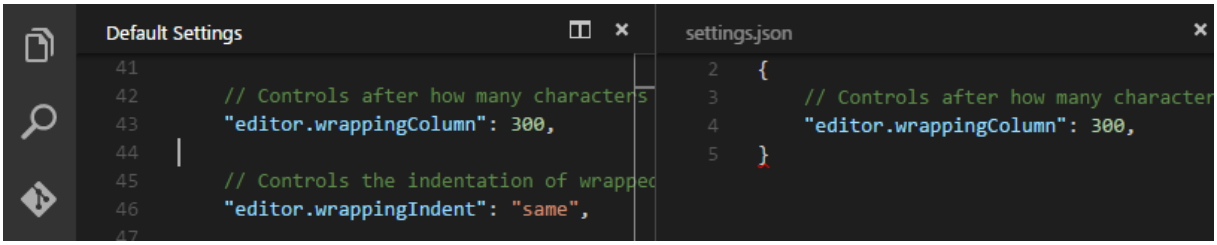
Viele Entwickler sind englischsprachige Programmierumgebungen gewohnt, nutzen aber deutschsprachige Windows-Desktops. In einem solchen Fall wird auch Visual Studio Code deutschsprachig angezeigt. Um Visual Studio Code trotzdem in einer (unterstützten) Sprache Ihrer Wahl anzuzeigen, ändern Sie den **"locale"**-Eintrag in der Datei **locale.json**. Diese blenden Sie nach Betätigung von **F1** durch Auswahl des Befehls **Configure Language** (oder, soweit Sie sich in deutschsprachiger Umgebung befinden, **Sprache konfigurieren**) ein.

Hier tragen Sie als neben **"locale"** als Wert **"de-de"** für eine deutschsprachige Oberfläche oder **"en-us"** für eine US-amerikanische Oberfläche ein. Die Wirksamkeit der Änderung erfordert einen Neustart von VS Code.

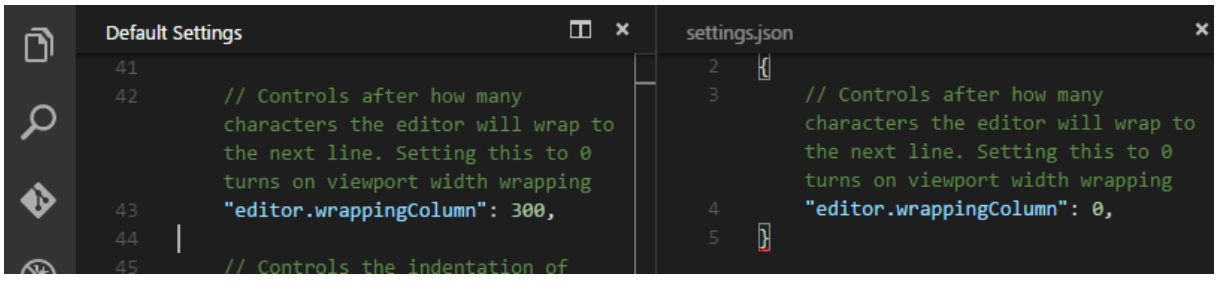
31. Tipp – Zeilenumbruch im Editor

Mit der Tastenkombination **Alt+Z** können Sie in einem Editorfenster dafür sorgen, dass kein Text rechts über den Rand des Editorfensters hinaus verschoben und damit unsichtbar wird.

Möchten Sie dauerhaft dafür sorgen, dass Texte im sichtbaren Bereich umgebrochen werden, so können Sie in den **settings.json** den Wert für **"editor.wrappingColumn"** verwenden. In den Standardeinstellungen beträgt der Wert 300, so dass Zeilen erst nach 300 Zeichen umgebrochen werden.



Doch welcher Wert eignet sich besser, wird doch das Editorfenster je nach Situation ständig in anderen Breiten angezeigt? Abhilfe schafft hier der Wert 0: Mit ihm erreichen Sie den dauerhaften Zeilenumbruch. Unmittelbar nach dem Speichern der neuen Einstellung zeigt sich der Effekt.



Visual Studio Code - Tipps & Tricks Vol. 1

1. Auflage - März 2016, 1. Aktualisierung im April 2016

© Copyright Microsoft Deutschland GmbH 2016
Alle Rechte vorbehalten. Dieses Dokument dient lediglich Informationszwecken.
Irrtümer und Fehler vorbehalten.

Dieses Dokument immer aktuell unter http://aka.ms/VSCodeTippsTricks

Autoren: Tobias Kahlert und Kay Giza

Kay Giza

- Audience Evangelism Manager bei Microsoft Deutschland
- Blog: http://www.giza-blog.de
- Twitter: http://twitter.com/KayGiza
- XING: Kay Giza
- LinkedIn: Kay Giza

Tobias Kahlert

- Bis März 2016 Werkstudent im Visual Studio Audience Marketing-Team bei Microsoft Deutschland
- Twitter: https://twitter.com/CubeCode
- Microsoft TechWiese.de Profil: Tobias Kahlert
- XING: Tobias Kahlert