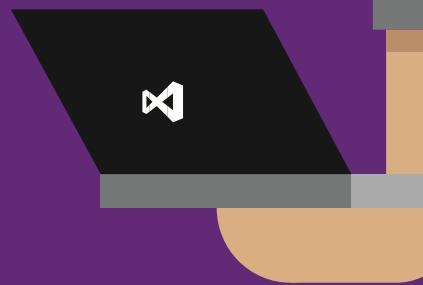


# Visual Studio Code Tips & Tricks Vol. 1



1st Edition – March 2016, Revision 1 (April 2016)

© Microsoft 2016

All rights reserved. This document is for informational purposes only.

Microsoft Deutschland GmbH · Konrad-Zuse-Str. 1 · D-85716 Unterschleißheim  
Tel. +49 (0)89 31760 · [www.microsoft.com](http://www.microsoft.com) · [www.techwiese.de](http://www.techwiese.de)

Authors: Tobias Kahlert and Kay Giza · Microsoft Germany

Editor: Mathias Schiffer

Localization: textoso · [www.textoso.com](http://www.textoso.com)

This book expresses the authors' views and opinions.

This document always up-to-date at: <http://aka.ms/VSCoTipsTricks>

# Contents

Visual Studio Code? .....	4
Preface.....	5
What is Visual Studio Code?.....	6
Tip 1 – Getting the Latest and Greatest as a VS Code Insider .....	6
Tip 2 – Multiple Cursors .....	8
Tip 3 – Using the Command Palette to Control VS Code .....	8
Tip 4 – Selecting a Language for a File .....	9
Tip 5 – AutoSave .....	10
Tip 6 – Hiding Undesired Folders .....	10
Tip 7 – Hiding Search Results.....	11
Tip 8 – Re-Inserting Regex Matches .....	11
Tip 9 – Emmet Snippets.....	12
Tip 10 – Keyboard Shortcut Chords.....	13
Tip 11 – Markdown Preview.....	14
Tip 12 – Visualizing CSS Selectors.....	14
Tip 13 – Latest Version of Dependencies in package.json .....	15
Tip 14 – Git Quick Change Info .....	15
Tip 15 – Refreshing Git-Views .....	16
Tip 16 – Git Inline Comparison .....	17
Tip 17 – Comparing Files .....	17
Tip 18 – Extensions.....	18
Tip 19 – Automatic Task Discovery for Gulp, Grunt and Jake .....	19
Tip 20 – Configuring PHP for Visual Studio Code .....	19
Tip 21 – Command Line Parameters .....	20
Console Support for Linux .....	20
Tip 22 – Storage Locations .....	21
Settings, Keyboard Shortcuts and Code Snippets .....	21
Extensions.....	21
Tip 23 – Synchronizing Settings and Extensions.....	21
Tip 24 – Deactivating Crash Reports .....	22
Tip 25 – Deactivating Telemetry.....	22

Tip 26 – Navigating Files in the Editor .....	22
Tip 27 – Opening Files in Dedicated Editor Windows .....	22
Tip 28 – Customizing Keyboard Shortcuts.....	23
Tip 29 – Authors’ Favorite Keyboard Shortcuts .....	24
Tip 30 – Changing the UI Language .....	24
Tip 31 – Enforcing Word Wrap in the Editor .....	25

# Visual Studio Code?

[Visual Studio Code](#) provides developers with a new choice of developer tool that combines the simplicity and streamlined experience of a code editor with the best of what developers need for their core code-build-debug cycle. Visual Studio Code is the first code editor, and first cross-platform development tool - supporting OS X, Linux, and Windows - in the [Visual Studio family](#).

At its heart, Visual Studio Code (VS Code) features a powerful, fast code editor great for day-to-day use. The Beta release of Code already has many of the features developers need in a code and text editor, including navigation, keyboard support with customizable bindings, syntax highlighting, bracket matching, auto indentation, and snippets, with support for dozens of languages.

For serious coding, developers often need to work with code as more than just text. Visual Studio Code includes built-in support for always-on IntelliSense code completion, richer semantic code understanding and navigation, and code refactoring. In the Preview, Code includes enriched built-in support for ASP.NET Core development with C#, and Node.js development with TypeScript and JavaScript, powered by the same underlying technologies that drive Visual Studio. Code includes great tooling for web technologies such as HTML, CSS, Less, Sass, and JSON. Code also integrates with package managers and repositories, and builds and other common tasks to make everyday workflows faster. And VS Code understands Git, and delivers great Git workflows and source diffs integrated with the editor.

But developers don't spend all their time just writing code: they go back and forth between coding and debugging. Debugging is the most popular feature in Visual Studio, and often the one feature from an IDE that developers want in a leaner coding experience. Visual Studio Code includes a streamlined, integrated debugging experience, with support for Node.js debugging and more to come later.

Architecturally, Visual Studio Code combines the best of web, native, and language-specific technologies. Using the [GitHub Electron Shell](#), Visual Studio Code combines web technologies such as JavaScript and Node.js with the speed and flexibility of native apps. Visual Studio Code uses a newer, faster version of the same industrial-strength HTML-based editor that has powered the "Monaco" cloud editor, Internet Explorer's F12 Tools, and other projects. And Visual Studio Code uses a tools service architecture that enables it to use many of the same technologies that power Visual Studio, including Roslyn for .NET, TypeScript, the Visual Studio debugging engine, and more. Visual Studio Code includes a public extensibility model that lets developers build and use plug-ins, and richly customize their edit-build-debug experience.

If you prefer a code editor-centric development tool or are building cross-platform web and cloud applications, we invite you to try out Visual Studio Code and let us know what you think!

# Preface

Less than [five years](#) ago we started developing the components for what was to become Visual Studio Code. Right from the start we were aware that if we wanted to build a tool for developers, we must aggressively use it ourselves.

It took us only four months before we started to do all our development using the predecessor of Code. There were many limitations and it was pretty rough. For example, in the beginning we did not have mouse support. As a team we used the tool daily – something we call “eating your own dogfood”, this helped us to tune and improve the tool continuously.

Since we released a first beta of Code in May 2015, a vibrant community has formed with the shared goal of improving Code. We keep receiving feedback in bug reports, tweets, questions on Stack Overflow, and we try to react to this feedback in our monthly releases.

[Tobias Kahlert](#) and [Kay Giza](#) (from Microsoft Germany) were two of the early users of Code, and they have been helping us answer questions from the community for some time now. This eBook is the next logical step in that process. In this collection of tips and tricks, the authors share their experiences using Code with you.

My goal is that our users can use Code as productively as the core development team does in their daily coding work. In this book, Tobias and Kay describe dozens of tips and tricks that will help you to get closer to this goal.

Happy Coding!

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

## About Erich Gamma

[Erich Gamma](#) is a Microsoft Distinguished Engineer and leads the Visual Studio Code project. He and his team work out of a small development lab in Zurich. The team is also responsible for the on-line Monaco editor, which is used in many Microsoft products. Previously he worked for IBM and he was one of the fathers of Eclipse and was leading the Eclipse Java development tools. Erich is also a member of the Gang of Four, which is known for its classical book, “Design Patterns - Elements of Reusable Object-Oriented Software.” Erich has collaborated with Kent Beck on developing JUnit, the de facto standard testing tool for Java software.

# What is Visual Studio Code?

Visual Studio Code (VS Code) is a free open source **code editor** for development and debugging modern cloud and web applications which is available for free on Linux, OS X and Windows. VS Code supports more than 30 different programming, markup and database languages, some of which are JavaScript, C#, C++, PHP, Java, HTML, R, CSS, SQL, Markdown, TypeScript, Less, Sass, JSON, XML and Python.

The lightning fast editor does not only satisfy developers with integrated debugging (including ASP.NET 5 and Node.js applications): Even Git support is available by the push of a button. Numerous Keyboard Shortcuts take care of your productivity at work.

In contradiction to Visual Studio 2015 and its predecessors, Visual Studio Code does not work with project files, but with files and folders. The environment can flexibly be enriched through the use of [Extensions](#) that remain independent from the underlying operating system. Code genuinely qualifies as the **perfect addition to the preferred development tools**.

Thanks to the integrated update functionality, Visual Studio Code is always up-to-date. Monthly update cycles reflect a dynamic advancement that is being driven in parallel for all supported platforms. And for [Visual Studio Code Insiders](#), there is even a feature preview edition available that can safely and securely coexist with the regular build on the same machine.



## A couple of notes about the eBook

This eBook was released to the public in March 2016. The development team is enhancing and extending Visual Studio Code day by day and upgrading existing installations on a monthly schedule. Even for an eBook, it's hard to keep pace with that. That's why based on your feedback, we are planning to [transfer this book to GitHub](#), so that you, our valuable readers, can contribute to existing and new tips for Visual Studio Code. Please tell us what you think.

If you're a Mac or a Linux user, we would also like to bring to your attention that this book has been written based on Windows and its key bindings. Kindly find [all keyboard shortcuts for your operating system](#) at code.visualstudio.com.

Finally, we'd like to suggest you visit the VS Code team's own and [latest collection of tips and tricks for Visual Studio Code](#), as you can never know enough tips and tricks. You may also want to check Valerii Iatsko's [repository with code packages and resources for VS Code](#): We recommend it!

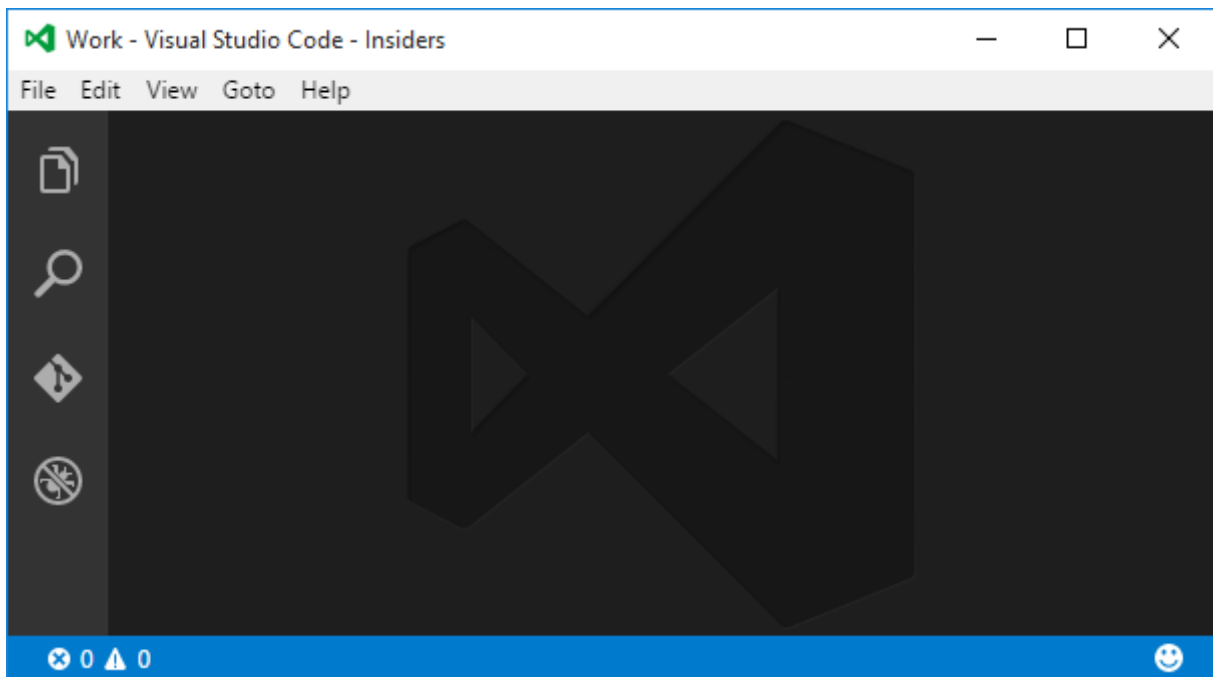
## Tip 1 – Getting the Latest and Greatest as a VS Code Insider

Visual Studio is an open source project under steady development. Even though changes are being documented in a public repository at <https://github.com/Microsoft/vscode>, it takes its time before new features have made it to the next public release.

If you want to work with prerelease versions of VS Code, you can additionally install the Insiders build. It is an isolated application with its own settings and extensions coexisting with the regular build on your machine. This allows you to test the latest features but still switch to the public release in case anything is malfunctioning.

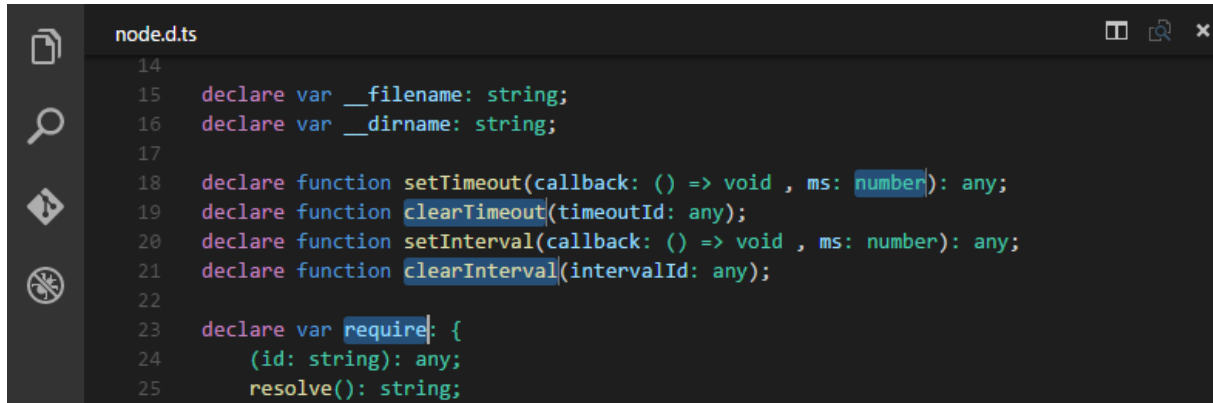
You can download the Insiders build from <https://code.visualstudio.com/insiders>.

#VisualStudioCodeInsiders



## Tip 2 – Multiple Cursors

Visual Studio lets you work on different sections of a file at the same time. This can come in handy when you need to perform many identical changes. To create a new cursor, simply click on the



```
node.d.ts
14
15 declare var __filename: string;
16 declare var __dirname: string;
17
18 declare function setTimeout(callback: () => void , ms: number): any;
19 declare function clearTimeout(timeoutId: any);
20 declare function setInterval(callback: () => void , ms: number): any;
21 declare function clearInterval(intervalId: any);
22
23 declare var require: {
24     (id: string): any;
25     resolve(): string;
```

respective area in the text editor while holding down the **Alt** key. From that point onwards any keyboard input will be applied to all cursor positions. It is also possible to select multiple sections in your code.

## Tip 3 – Using the Command Palette to Control VS Code

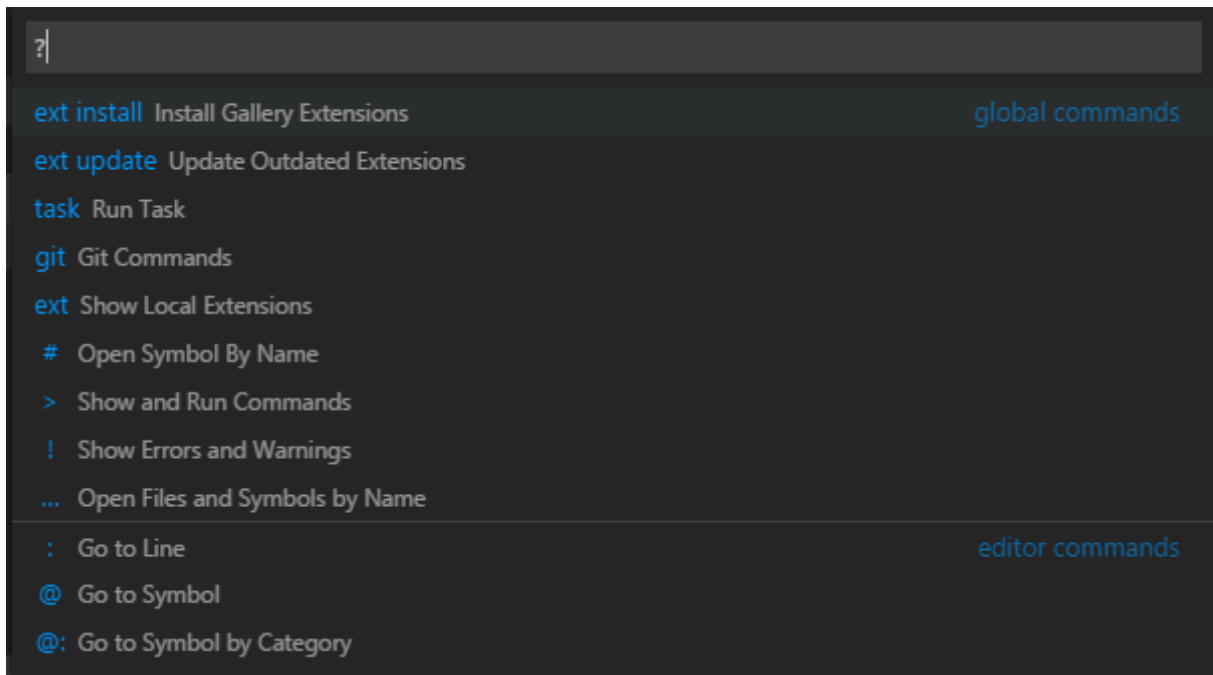
When using Visual Studio Code, you will scarcely be able to avoid the commands palette. It is usually being accessed by pressing **F1** to enable command entries. Still, it can do a lot more.

If you watch closely, you will notice the „>“ in the command line after pressing **F1**. This is because the first letter or word will determine the input mode. Starting with „>“ sets the palette to command mode.

Press **Ctrl+P** if you want to open the palette without any preset input, which gets you into navigation mode: It initially displays a list of open files. Entering a couple of letters filters the opened files and the files in opened folders for display. This is an easy means of quickly hopping between files. If you're within a project that supports IntelliSense, this functionality even searches detected symbols.

Additional modes allow you to control many areas of VS Code. You can display a list of all combinations by entering „?“ . It's really worth memorizing and using the various options. They often help you in keeping your hand away from the mouse, thus improving your productivity.

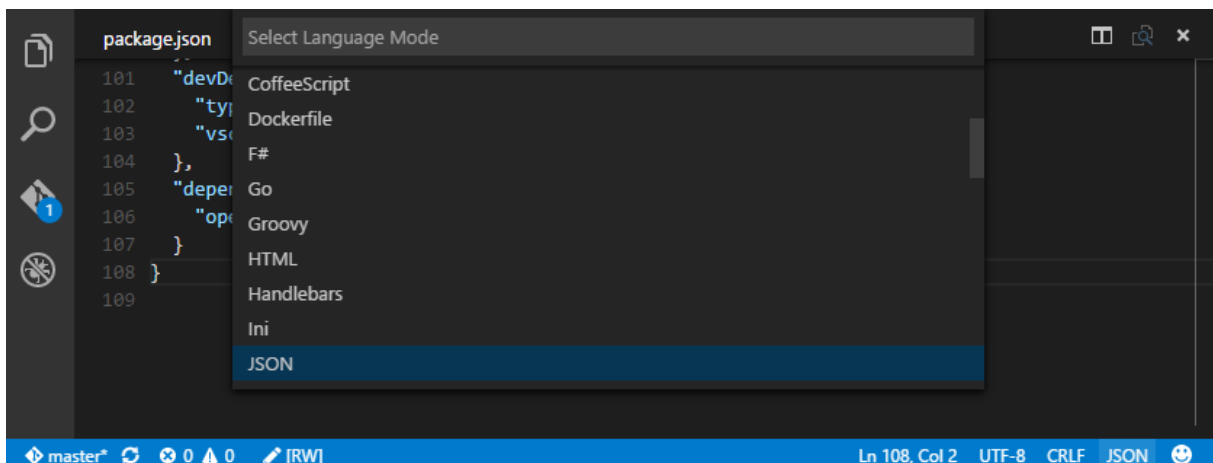




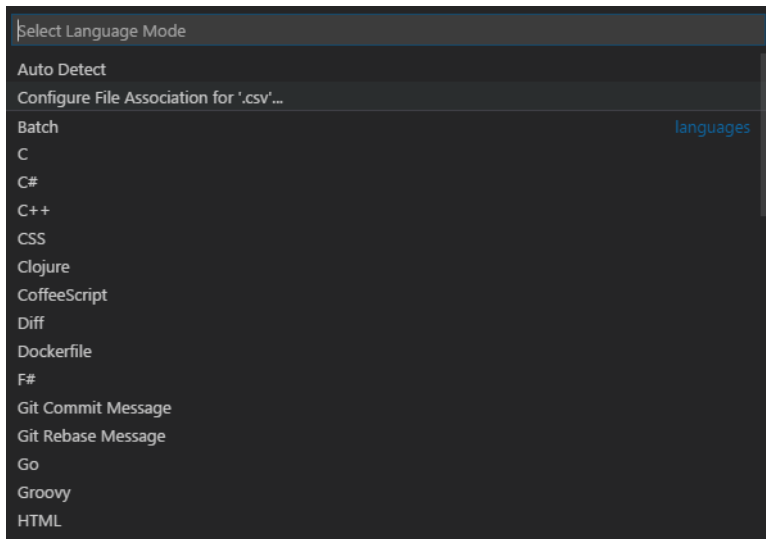
## Tip 4 – Selecting a Language for a File

Every now and then, VS Code might not recognize the language of an opened file. Oftentimes this happens when creating a new file or when a file name does not come with a file extension.

VS Code supports syntax highlighting for many programming languages. Not making use of this would be too bad. To select a language for the active file, either run the command **Change Language Mode** or click the statusbar's current language area in the bottom right corner, which will open a list of available languages to make a selection.



There is another case in which VS Code might not recognize the language for a file: That is if no language has been associated to the file name suffix. You can create the lacking association in the Select Language Mode list.



## Tip 5 – AutoSave

VS Code supports AutoSave. This feature automatically saves the current document after each change. This is especially useful for web related files. You do not need to click on Save and still all files are up-to-date when clicking refresh in the browser. Take care when using automated tasks that keep repeating when a file changes (e. g. for transpile TypeScript to JavaScript) as this can cause heavy CPU loads.

AutoSave can be activated and deactivated in the settings using the option **"files.autoSave"**. It is also possible to delay AutoSave operations for a specified amount of time by setting **"files.autoSave"** to **"afterDelay"** and setting **"files.autoSaveDelay"** to the desired delay in milliseconds, e. g.:

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

## Tip 6 – Hiding Undesired Folders

If you have files or folders that you'll never edit within opened folders, you can hide those files or folders in the explorer view. One example would be the **node\_modules** folder in a Node.js project.

To hide the file, folder or any pattern, you need to reference them in the **"files.exclude"** option in your global or workspace settings. Use the workspace settings to hide those entries only within the active project.

```
settings.json .vscode
1 // Place your settings in this file to overwrite default and user settings.
2 {
3   "files.exclude": {
4     "out": true, // set this to true to hide the "out" folder with the compiled JS files
5     "node_modules": true
6   },
7 }
```

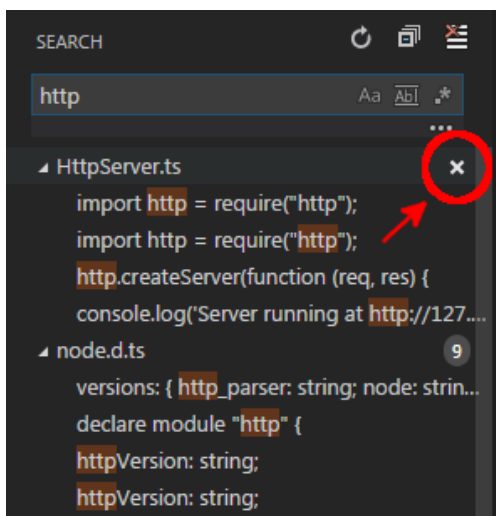
Opening the workspace settings is as easy as executing the command **Preferences: Open Workspace Settings**. If there is no workspace settings file yet, VS Code will create it automatically.

One useful example of pattern use is hiding any JavaScript resources generated from TypeScript:

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

## Tip 7 – Hiding Search Results

Visual Studio offers a neat search function incorporating all active files. When iterating through the search results it can be helpful to hide results from that list for files that you already checked. If areas within the found files need to be modified, this can help keeping an eye on areas that you already touched. Beneath each filename, VS Code displays the number of search hits within that file. If you hover that number with your mouse pointer, you'll see a cross appear which you can click to hide that filename from the list.



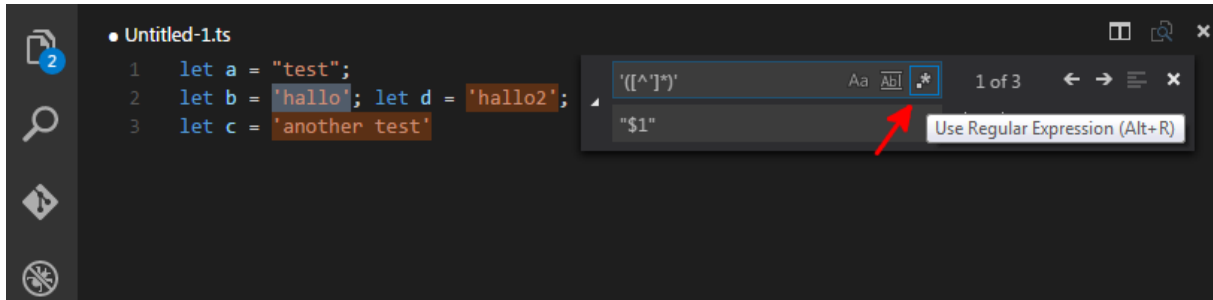
All files will be listed again after executing another search.

## Tip 8 – Re-Inserting Regex Matches

Visual Studio Code's search functionality supports regular expressions. This allows you to not only find text, but also entire syntactical constructs. Its real power however lies in combining it with the replace function, which does not only insert static text, but can also relate to the results of the regular expression search.

Use brackets to define groups within regular expressions. The text fragments being found for these groups can be referred to using **\$1**, **\$2**, **\$3**, ... in the text to be replaced.

If, for example, you want all string expressions to be embraced by double quotes, enter **017** in the search textbox and **"\$1"** in the replace textbox. The first expression finds all string expressions embraced by single quotes, with the content between the quotes being defined as group 1. The second expression replaces the entire text fragment with two double quotes and the content of the original string expression (the value of group 1) in between.



Note: If you want to use the dollar sign as a term, simply use two of them subsequently (\$\$).

## Tip 9 – Emmet Snippets

In spite of their good readability, HTML and XML are often hard to write. To accelerate the creation of such documents, VS Code supports Emmet snippets. Emmet snippets are expressions similar to CSS which are being converted to e. g. HTML or XML after pressing the tab key.

In its easiest form, just one element would be created. If, for example, you write „**html**“ to a file that VS Code recognizes as an HTML file, and you press the tab key twice after that, this text is being replace with „**<html></html>**“. The cursor is being positioned between the two tags.

But Emmet snippets can do more. There is a group of constructs that are being supported by Emmet snippets: If you press the tab key after entering „**li\*5**“, VS Code will insert five „**<li></li>**“ fragments. Find all sorts of Emmet constructs at <http://docs.emmet.io/cheat-sheet/>

```
● emmet.html
1  html>(head>title{Emmet Snippet Test})+(body>h1+li*3)|
```



```
● emmet.html
1  <html>
2  <head>
3  <title>Emmet Snippet Test</title>
4  </head>
5  <body>
6  <h1></h1>
7  <li></li>
8  <li></li>
9  <li></li>
10 </body>
11 </html>
```

Other than HTML and XML, VS Code also supports additional formats, currently this includes Razor, CSS, Less, SCSS, XSL, Jade, Handlebars, HBS, JSX and TSX.

## Tip 10 – Keyboard Shortcut Chords

Just like Visual Studio, Visual Studio Code supports keyboard shortcut chords: The editor is awaiting a second keyboard shortcut after the first keyboard shortcut has been entered. This allows for grouping of similar commands.

As an example, you could change the search in a file from **Ctrl+F** to **Ctrl+F Ctrl+F** and the global search in all files to **Ctrl+F Ctrl+G**. In **keybindings.json**, the keyboard shortcuts for the chord must be separated using a blank space.

```
keybindings.json C:\Users\Mathias Schiffer\AppData\Roaming\Code - Alpha\User
1  // Platzieren Sie Ihre Tastenbindungen in dieser Datei, um die Standardwerte zu überschrei
2  [
3    {
4      "key": "ctrl+f ctrl+f",
5      "command": "actions.find",
6      "when": "editorTextFocus"
7    }
8  ]
```

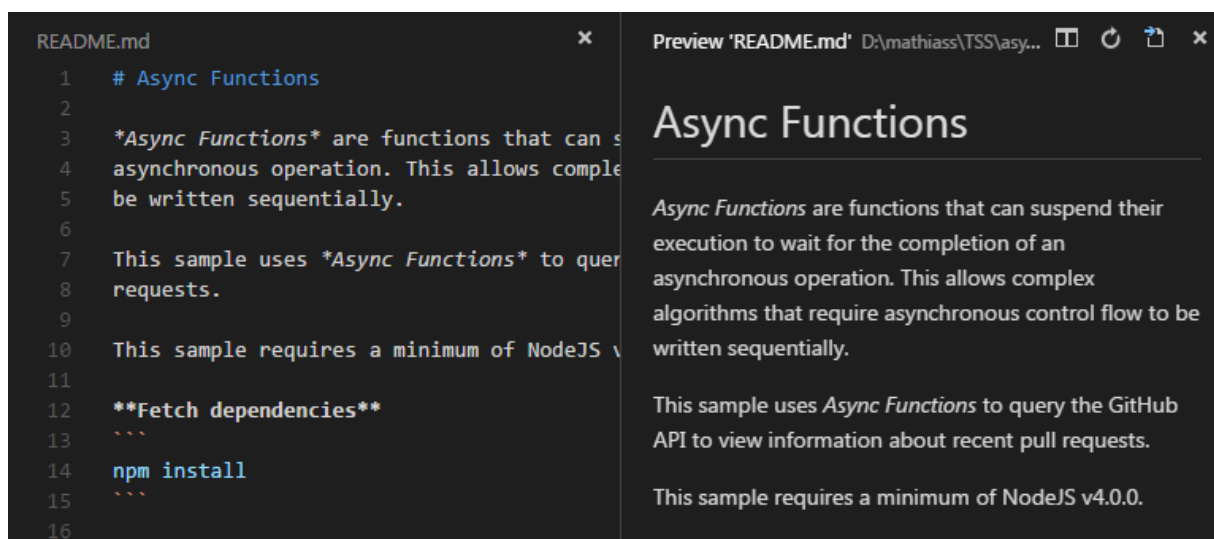
Define Keybinding (Ctrl+K Ctrl+K)

Visual Studio Code is awaiting additional input after the first shortcut of a chord has been applied. You'll find this referenced in the status bar as „(Ctrl+F) was pressed. Waiting for second key of chord“.

## Tip 11 – Markdown Preview

Markdown is a simple markup language for authoring formatted text. Even if markdown is well legible in its original shape, it is a lot easier to read when it is being displayed as a properly formatted document, especially regarding external resources like images and such.

Visual Studio offers a markdown preview function that can be opened beneath the text. VS Code is using the GitHub Flavoured Markdown which offers additional functions over the original markdown format. This comes in handy especially for GitHub “read me” documents.



Using **Ctrl+Shift+V**, you can switch between Markdown and Preview. The chord **Ctrl+K, V** opens the preview to the side. **Alt+Z** toggles word wrapping, which can also be done using the **View** menu.

## Tip 12 – Visualizing CSS Selectors

CSS selectors precede any CSS clause. They define what HTML elements to style with the subsequent properties. Selectors can become pretty complicated – and be it because sometimes you just might not be too familiar with one of them that you are using less frequently.

Visual Studio Code solves this problem by presenting a preview for the current selector that shows what HTML elements would have to be like to be covered by the selector.

```

1 <section class="article">
2   <p>
3
4 section.article > p {
5   color: green;
6 }

```

```

1 <h1>
2 <p>
3   <color important="true">
4     ...
5     <element>
6 h1 + p > color[important=true] * {
7   color: red;
8 }

```

```

1 <p id="MainText">
2
3 p#MainText {
4   color: green;
5 }

```

```

1 <section class="footer">
2
3   ...
4   <a :hover>
5 section.footer a: hover {
6   text-decoration: underline;
7 }

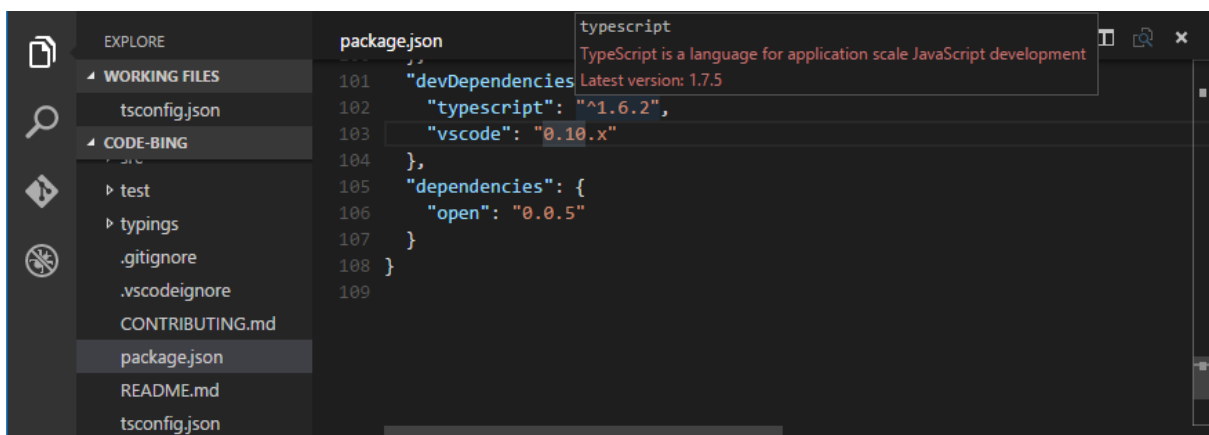
```

### Tip 13 – Latest Version of Dependencies in package.json

When working on a Node.js project, **package.json** contains a couple of sections that define dependencies. For each dependency, one or more version numbers have to be specified.

VS Code allows you to display module information. Simply move the mouse pointer above one of the dependencies to display the name, a short description and the latest version number for that module.

You can add new dependencies for modules using **Ctrl+Spacebar**.



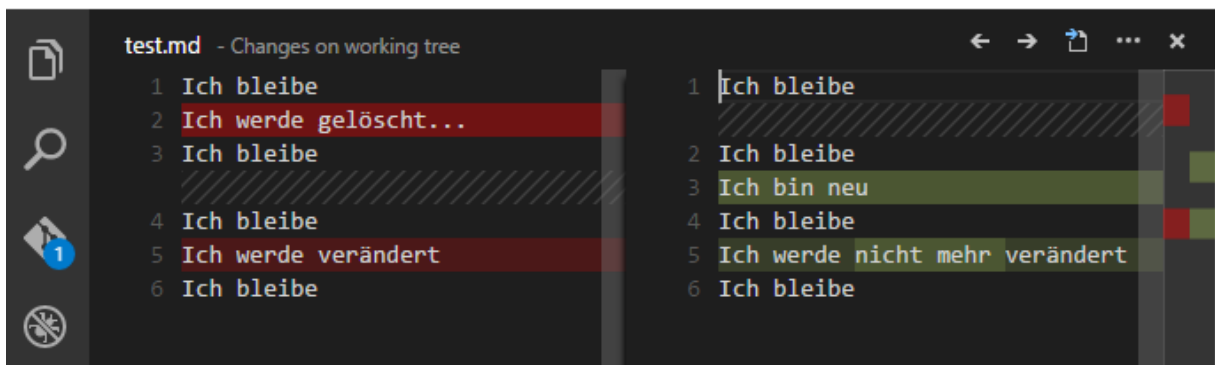
### Tip 14 – Git Quick Change Info

When working with a Git repository, VS Code offers a neat way of comparing the version you're working on with the current one in the repository using Git Compare View.

VS Code even shows code changes when simply editing a file that is under Git source control. At the left hand side of the editor, right between row numbers and code, you'll find indicators for changed, deleted and new rows.

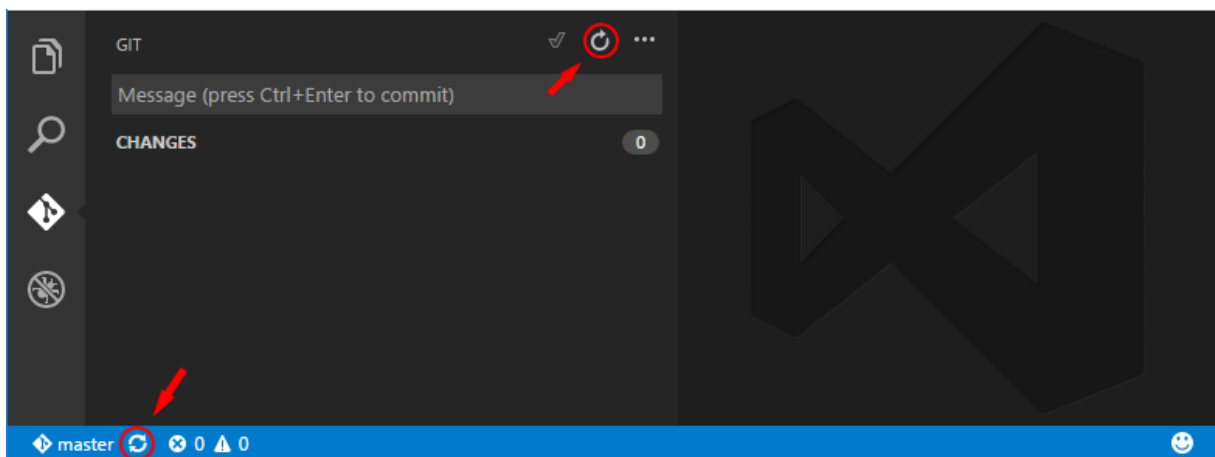
- Green line: Rows with green markers have been added
- Blue line: Rows with blue markers have been altered
- Red arrow: A red arrow indicates a row has been deleted

You can change to Changes View by clicking the **Switch To Changes View** icon.



## Tip 15 – Refreshing Git-Views

Visual Studio Code makes use of the external program Git to track changes in a public Git directory. Sometimes you'll face a view that's not entirely up-to-date or even cluttered. To refresh a messy view, use the refresh icons located at the Git header or at the status bar.

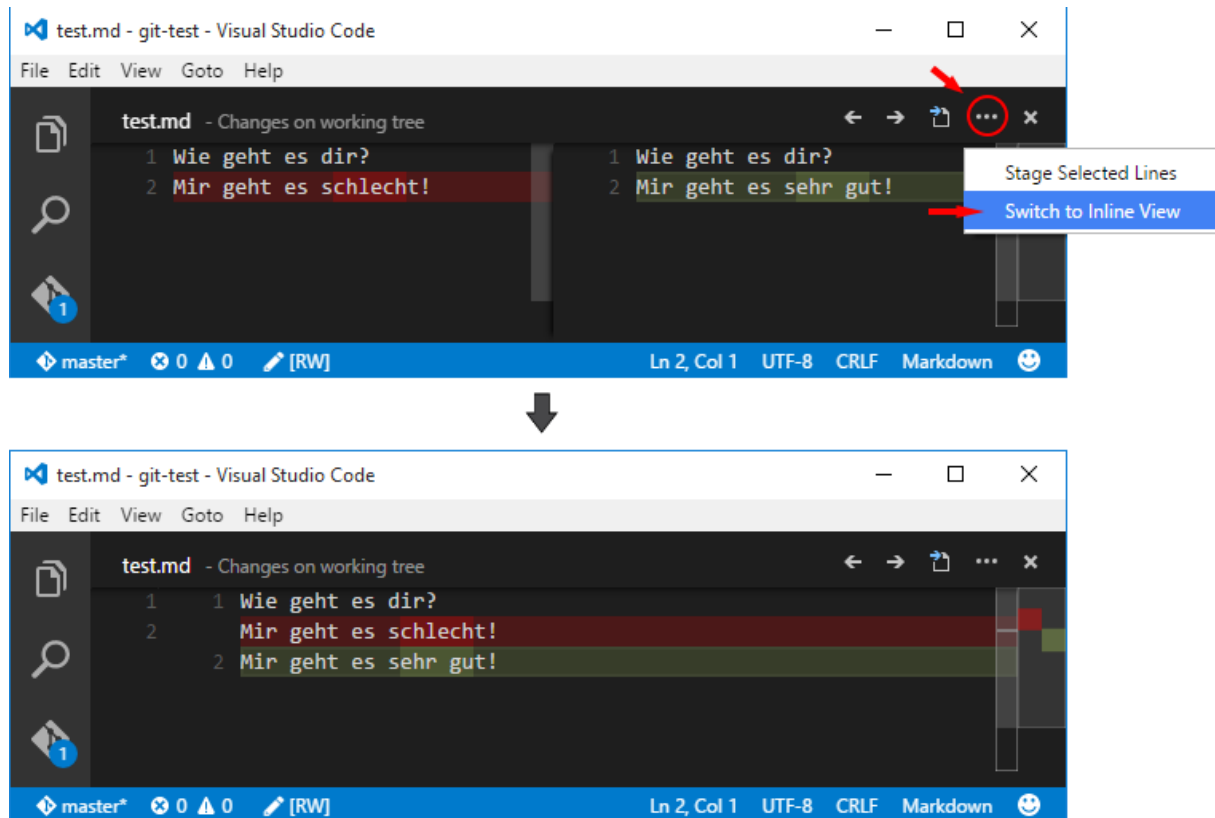




## Tip 16 – Git Inline Comparison

Having Visual Studio Code display changes in a file as compared to the current Git version opens a view which displays the changes side-by-side.

But Visual Studio also offers a way to display all the changes in a single code window: Use the sub menu **Switch to Inline View** from the header's "... " menu to get to the Git inline comparison.



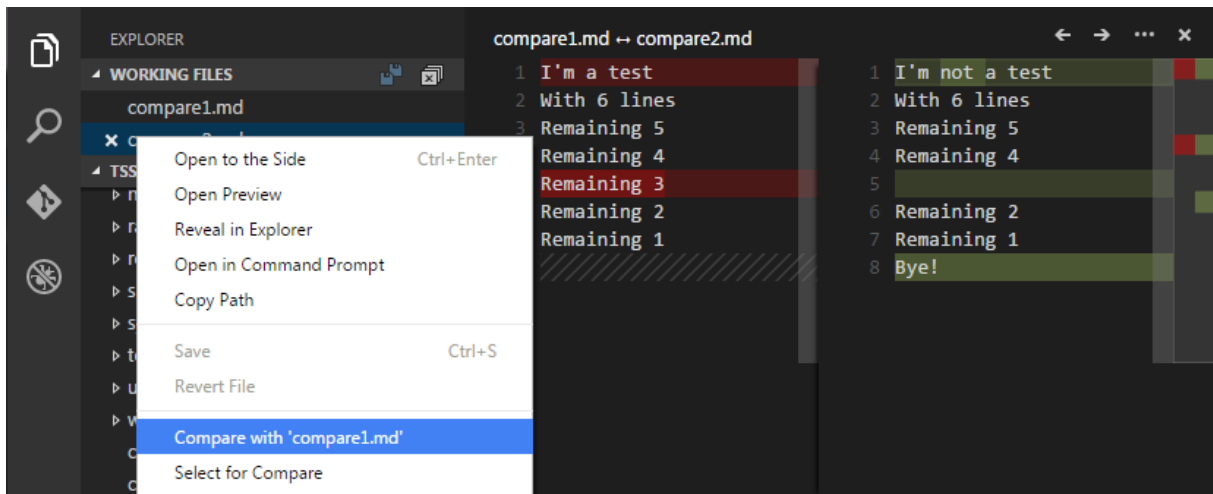
Simply use the same menu item again to return to side-by-side view.

## Tip 17 – Comparing Files

VS Code offers a great viewport for comparing two files. You'll often get this view when using Git with VS Code. If you click on a file name within the Git area, all alterations to the previous file are being displayed.

But VS Code can do more than that. It can actually compare any file to any other file. You can launch the comparative view using the Command Palette or the explorer area's context menu. For the Command Palette, you execute **Files: Compare Active File With...** and choose a file that you wish to compare to the currently opened file. To take the context menu route, activate the context menu for a file in explorer view and click on **Select for Compare**. If you subsequently activate the context menu

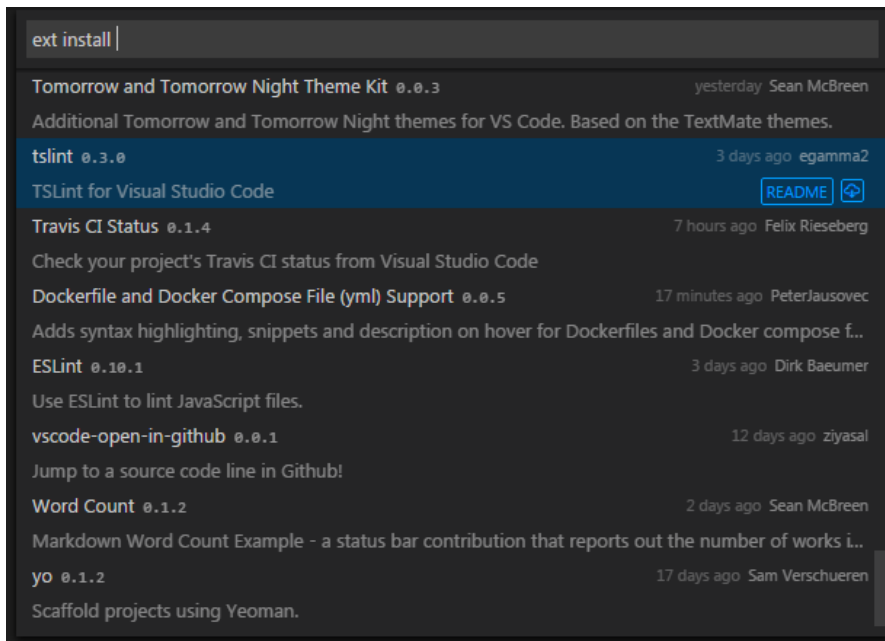
for another file, you can click on **Compare with ,<First File>'**. Any of the two options launches the mparison view.



## Tip 18 – Extensions

Visual Studio Code comes with a built-in bunch of functionality. But the best of all features is to create new functionality and easily share it with the rest of the world: Extensions. Extensions are being developed with TypeScript or JavaScript, they can be shared in the Visual Studio Code Marketplace. Other users can then search the Marketplace and install available extensions with no more than a few mouse clicks. This way, VS Code can be extended by third parties.

All Extensions can be checked at <https://marketplace.visualstudio.com/#VSCode> and are ready to be installed by simply executing **Extension: Install Extension**.



Additional information about Extensions is available at <https://code.visualstudio.com/Docs/editor/extension-gallery>. The neatest thing about Extensions is

than you can create them yourself. We have detailed information on that, including developer guidance and samples, at <https://code.visualstudio.com/Docs/extensions/overview>.

## Tip 19 – Automatic Task Discovery for Gulp, Grunt and Jake

Tasks enable Visual Studio Code to execute all kinds of external applications. To make them available from the Command Palette, tasks have to be listed in `.vscode/tasks.json` with their respective parameters and options.

If you're working on a Node.js project using Gulp, Grunt or Jake however, VS Code can automatically detect all tasks defined by those, releasing you from having to create a respective `tasks.json`. You can simply execute **Tasks: Run Task** to display a list of available tasks for you to choose from.



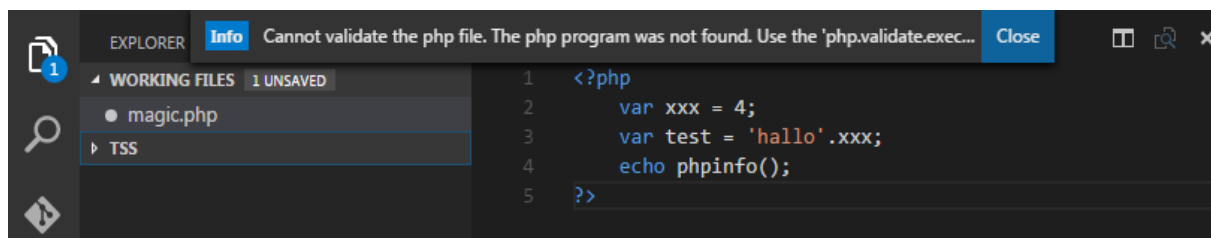
```
gulpfile.js
1
2
3
4
5
6 var gulp = require("gulp");
7
8
9 gulp.task('default', ['debug']);
10
11 gulp.task('build', function() {
12   // ..
13 });
14
15 gulp.task('debug', ['build'], function(done) {
16   // ...
17 });
```

task  
build  
debug  
default

## Tip 20 – Configuring PHP for Visual Studio Code

Visual Studio Code supports syntax highlighting for PHP, including help texts and parameter information for many of its functions.

But VS Code can also validate source code and display errors live. This requires that the path to a PHP interpreter be configured. You'll get an appropriate warning after opening a PHP file if this is not yet been done: **"Cannot validate the php file. The php program was not found. Use the 'php.validate.executablePath' setting to configure the location of 'php'"**.



```
EXPLORER Info Cannot validate the php file. The php program was not found. Use the 'php.validate.exec... Close
WORKING FILES 1 UNSAVED
  magic.php
  TSS
1 <?php
2   var xxx = 4;
3   var test = 'hallo'.xxx;
4   echo phpinfo();
5 ?>
```

If there is no PHP installed on your machine yet, you can download it from <http://php.net/downloads.php> and install it (i. e. copy the archive files) locally. The target folder should now contain `php.exe` (for Windows). You need to provide the path to this file as the value for Visual Studio Code's setting `"php.validate.executablePath"`.

Now opening a PHP file should result in the errors being displayed.

```
1 <?php
2 var xxx = 4;
3 var test = 'hallo'.xxx;
4 echo phpinfo();
5 ?>
```

Hint: To be able to debug PHP with VS Code, install the following Extension from the Marketplace: [PHP Debug adapter for Visual Studio Code](#).

## Tip 21 – Command Line Parameters

On install, Visual Studio Code is being added to the environment variable PATH so that you can start it from the console simply typing **code**. This comes in handy, among other examples, when you need to open code created by other programs. Just typing **code** results in an empty VS Code editor launching, though. The following lists available parameters to the call.

First of all, VS Code looks at any parameters as paths to files and folders. They are being opened in a new window of VS Code. Specifying multiple folders will result in multiple instances of VS Code. You can include specific options before, within or after path parameters. These are the most common ones:

- r, --reuse-window** Opens the specified files in an existing VS Code instance if possible (or launches a new one otherwise).
- n, --new-window** Forces VS Code to launch a new instance anyway.
- g, --goto** Files passed in the format „**PathToFile:Row:Column**“ are being opened with the cursor positioned at the specified row and column.
- disable-extensions** Launches VS Code without loading any Extensions.
- verbose** Have VS Code output debug information.
- diff <file1> <file2>** Open a diff view for the two files.

### Console Support for Linux

To be able to launch Visual Studio Code from a console in Linux you will need to create a symbolic link from **/usr/local/bin/code** to the VS Code executable. You can achieve this using the following command:

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

## Tip 22 – Storage Locations

Visual Studio saves files to different locations within the file system. It can come in handy to know what is being stored where.

### Settings, Keyboard Shortcuts and Code Snippets

VS Code Settings are easily accessed using the Command Palette. They are being stored in **settings.json**. Depending on what operating system you use, this file can be found at different locations. In this folder, you'll additionally find **keybindings.json** at this location, and also a folder named *snippets* which is the user storage containing code snippets for respective languages.

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

If you're using the Insiders build, go for `%APPDATA%\Code Insiders` or `.vscode-insiders`.

### Extensions

Extensions are being stored within a specific folder. Any Extension has its own sub folder named after the Extension's title and its author's name. Upon launch, Visual Studio Code searches this folder to load all the Extensions stored in it. That means installing an Extension requires no more than just moving or copying the Extension's folder to VS Code's Extensions folder:

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

## Tip 23 – Synchronizing Settings and Extensions

Visual Studio Code does not natively come with functionality that supports synchronizing settings and installed Extensions between machines. But as all settings are being stored in files and folders, it is sufficient to synchronize those.

Services like OneDrive or Dropbox can provide machine independent storage accessible from anywhere. An easy way is to replace the folders referred to in tip 22 with new, synchronized folders. Of course you would first move all settings and Extensions to those new folders. Creating the links for them can easily be done using the console:

- **Windows:** `mklink /d <VSCode-Folder> <Synchronized-Folder>`
- **Mac & Linux:** `ln -s <Synchronized-Folder> <VSCode-Folder>`

Another, even easier way to get the synchronization task done is using the Extension [Visual Studio Code Settings Sync](#) from the Visual Studio Marketplace which takes advantage of GitHub GIST.

## Tip 24 – Deactivating Crash Reports

When Visual Studio Code crashes, it creates a report to be sent to Microsoft in order to help improve the product. If you prefer not to create and transfer such a report, you can set the option value of **"telemetry.enableCrashReporter"** in your settings to **false**. VS Code needs to be restarted in order for the change to take effect.

## Tip 25 – Deactivating Telemetry

VS Code collects data about how it is being used. This data is being sent to Microsoft where it is anonymously being used to improve VS Code. If you prefer not to transfer such telemetric data, you can deactivate telemetry altogether in your settings. Find **"telemetry.enableTelemetry": true** and copy it to your desired **settings.json** with a modified value as **"telemetry.enableTelemetry": false**. VS Code needs to be restarted in order for the change to take effect.

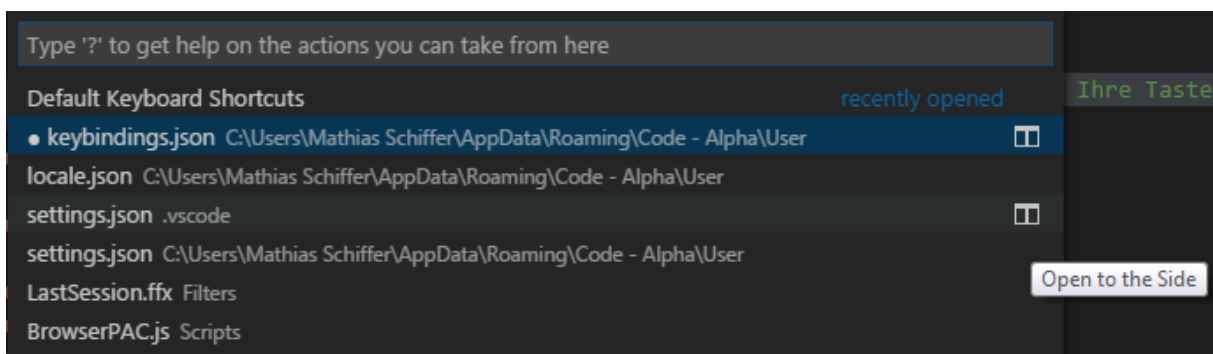
## Tip 26 – Navigating Files in the Editor

The VS Code editor is a powerful tool that shows its strength especially when using [keyboard shortcuts](#). Many of the existing shortcuts can be retrieved from VS Code's menu, where they are being listed right beside the menu item captions, and from the Command Palette after pressing **Ctrl+Shift+P**.

A special shortcut is **Ctrl+E** (which can be used synonymously with **Ctrl+Tab**): It opens up a list of recently used files which can be opened in the current code editor. Navigating this list is easily done by repetitive use of the shortcut while keeping the **Ctrl** key down. Additionally press the **Shift** key to traverse the list in opposite direction. As an alternative, you can also use the arrow keys on your keyboard while the **Ctrl** key remains down.

## Tip 27 – Opening Files in Dedicated Editor Windows

VS Code supports opening files in an additional editor window side-by-side to the existing one ("to the side"). To do so, simply hold down the **Ctrl** key while clicking on a file name in the explorer area. File lists come with an icon to the right that appears for selected items in the list, resulting in the same effect.



To open the definition of a selected term to the side, use the keyboard chord **Ctrl+K, F12**.

You can directly navigate to any of the up to three editor windows using **Ctrl+1**, **Ctrl+2** and **Ctrl+3**. You can close any of those windows using **Ctrl+W**.

To shift focus to the editor window and vice versa, e. g. from the explorer window or the search window and back, press **Ctrl+B**.

## Tip 28 – Customizing Keyboard Shortcuts

Using keyboard shortcuts makes you much more productive than frequently having to use your mouse, lifting your hand away from the keyboard. You'll find the most important Visual Studio Code keyboard shortcuts for your operating system at

<https://code.visualstudio.com/docs/customization/keybindings>.

Unless you're using a US keyboard layout, do carefully read that page's information about international keyboard layouts: While Visual Studio Code displays the right keyboard shortcuts for your keyboard layout, it is always using a US keyboard layout internally. This also applies to the file **keybindings.json** which defines the keyboard shortcuts. Where there's a difference between the keyboard layouts, you're going to see a blue circle with a white "i" in it. Hovering it with your mouse pointer will display a hint about the correct keys for your keyboard.

```
124 { "key": "ctrl+x", "command": "editor.action.clipboardCutAction" },
125 { "key": "s", "command": "editor.action.clipboardPasteAction" },
126 { "key": "c", "command": "editor.action.clipboardPasteAction" },
127 { "key": "ctrl+/", "command": "editor.action.commentLine",
128 "when": "editorTextFocus" },
```

The easiest way to create your own keyboard shortcut is to make use of the keyboard chord **Ctrl+K, Ctrl+K**. After entering your desired keyboard shortcut and pressing the **enter** key, you'll be presented a new **keybindings.json** entry for that shortcut. You'll still have to set values for **command** and **when** to define the behavior of your new keyboard shortcut.

Taking a look at the default keyboard shortcuts by clicking **File | Preferences | Keyboard Shortcuts** will provide helpful examples. Scroll down the file to discover loads of commands that have not yet been associated to keyboard shortcuts in the default VS Code setup. You may end up wanting to use some of those.

## Tip 29 – Authors’ Favorite Keyboard Shortcuts

Out of the many keyboard shortcuts available, the authors have found that they are continuously using a good couple of them to be accelerate their tasks. Some of them are (for Windows):

- **Alt+Z** Toggle Word Wrap
- **Ctrl+P** Quick Open
- **Ctrl+Tab** Navigate Recent Files
- **Ctrl+1/2/3** Focus First/Second/Third Editor
- **Ctrl+W** Close Active Editor
- **Ctrl+/** Comment in/out Line(s)
- **Ctrl+Shift+K** Delete Line(s)
- **Ctrl+Click** Open File from Explorer to the Side
- **Alt+C** Find Case Sensitive
- **Alt+W** Find Whole Word
- **Ctrl+K, F12** Open Declarations to the Side
- **Ctrl+D** Add Selection to next Find Match
- **Ctrl+K, Ctrl+D** Move Selection to next Find Match
- **Alt+Down** Move Line Down
- **ALT+Up** Move Line Up

## Tip 30 – Changing the UI Language

Visual Studio Code’s UI language adheres to your desktop settings – provided those are locale settings supported by VS Code (if not so, the UI will be displayed using the US locale).

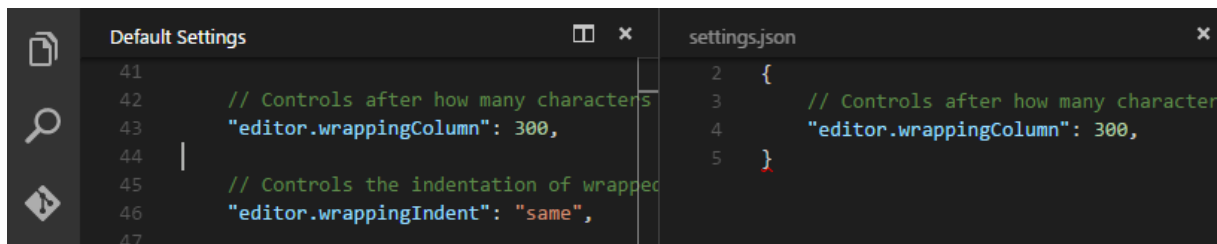
Many developers outside the US are accustomed to using US-English UIs in their development environments while having their desktops configured to use their local language and settings. You may have additional or other reasons to switch the VS Code UI language. Fortunately, you do not need to switch your entire desktop to the desired regional settings, as VS Code comes with a setting that controls its UI language.

To launch Visual Studio Code in a (supported) language of your choice, edit **locale.json** which you can open by pressing F1 and executing **Configure Language**. Set the value for **"locale"** to the desired locale ID (**"en-us"** for the US, **"de-de"** for Germany, **"fr-fr"** for France etc.). VS Code needs to be restarted in order for the change to take effect.



## Tip 31 – Enforcing Word Wrap in the Editor

Using **Alt+Z** in an editor toggles Word Wrap, making sure you're not missing any text in long lines that extend beyond your editor's visible area. But what if you want that setting to be permanent?



The screenshot shows two side-by-side windows in VS Code. The left window, titled 'Default Settings', shows the following code:

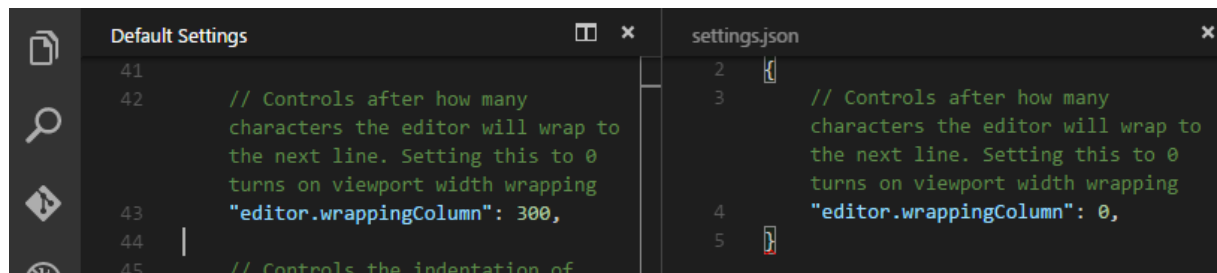
```
41
42 // Controls after how many characters
43 "editor.wrappingColumn": 300,
44 |
45 // Controls the indentation of wrapped
46 "editor.wrappingIndent": "same",
47
```

The right window, titled 'settings.json', shows the following code:

```
2 {
3 // Controls after how many character
4 "editor.wrappingColumn": 300,
5 }
```

VS Code has a setting in **settings.json** that takes care of line length in the editor: The default value for **"editor.wrappingColumn"** is 300, meaning a long line will not wrap before exceeding 300 characters.

What value is more suitable to prevent text from being hidden in long lines, given that your editor windows differ in width in different scenarios? The solution is using 0 as a value: It takes care of permanent word wrapping in your editor. Saving the file immediately activates the new setting.



The screenshot shows two side-by-side windows in VS Code. The left window, titled 'Default Settings', shows the following code:

```
41
42 // Controls after how many
43 characters the editor will wrap to
44 the next line. Setting this to 0
45 turns on viewport width wrapping
46 "editor.wrappingColumn": 300,
47 |
48 // Controls the indentation of
```

The right window, titled 'settings.json', shows the following code:

```
2 {
3 // Controls after how many
4 characters the editor will wrap to
5 the next line. Setting this to 0
6 turns on viewport width wrapping
7 "editor.wrappingColumn": 0,
8 }
```

## Visual Studio Code - Tips & Tricks Vol. 1

1st Edition – March 2016, Revision 1 (April 2016)

© Copyright 2016 - Microsoft Deutschland GmbH

All rights reserved. This document is for informational purposes only.

This book expresses the authors' views and opinions.

This document always up-to-date at: <http://aka.ms/VSCoDeTipsTricks>

Authors: Tobias Kahlert and Kay Giza

### Kay Giza



- [Audience Evangelism Manager](#)  
Microsoft Germany
- Blog: <http://www.giza-blog.de>
- Twitter: <http://twitter.com/KayGiza>
- XING: [Kay Giza](#)
- LinkedIn: [Kay Giza](#)

### Tobias Kahlert



- Internship at the Visual Studio Audience Marketing-Team Microsoft Germany until March 2016
- Twitter: <https://twitter.com/CubeCode>
- Microsoft TechWiese.de Profile: [Tobias Kahlert](#)
- XING: [Tobias Kahlert](#)