# DOWNLOAD PDF SUBLIME TEXT POWER USER

## Chapter 1 : My Text Editor Journey: Vim, Spacemacs, Atom and Sublime Text - Tristan Hume

*Ever wonder how so many great developers seem to get so much done? You probably aren't getting enough out of your text editor. Invest in your text editor skillset.*

My Text Editor Journey: Most detailed is my reasoning for abandoning Spacemacs , despite being a top contributor and power user, although many of my criticisms of Vim also apply to Spacemacs and vice versa. Back then I only used the basics: But, I heard tell of the true power one gained upon learning to use a real editor like Vim or Emacs. I watched screencasts where Vim masters would perform impressive editing operations in a couple keystrokes. I learned the keyboard shortcuts with vimtutor and printed cheat sheets. I read tons of blog articles often conflicting on learning and using Vim the right way. I tried using a blank. However, this was taking far too long, my editor was missing key functionality from Sublime and Textmate like a file tree, good autocomplete, open in project, and support for languages I used. It was also ugly. So I started using the spf13 Vim distribution. It was nice, and had most of the features I wanted. You can still find my modified spfbased vimrc here. I was reasonably happy with this setup and continued using it for over 6 months. However, there were many pain points. For example, my tab key was bound to tons of different things like autocomplete, snippet expansion, indentation, moving between snippet fields and inserting the literal tab character. Many of those overrode each other in different contexts, but very often it chose the wrong one. Even after I fixed it, the hours I spent diagnosing the issue, figuring out how to resolve the conflicts, implementing it, then re-learning my muscle memory probably erased weeks of sub- second Vim speed gains. Another issue I had was that Vim was mouse-hostile. I was fully aware that the Vim philosophy is to just never use the mouse. However, even with plugins like EasyMotion and ideal vim shortcut use the keyboard is slower for some selection tasks like selecting a range of text far from the cursor than the mouse is. Often using Vim shortcuts felt faster because my brain was engaged figuring out the optimal combination of motions and looking for EasyMotion hints, but whenever I timed myself I was consistently much slower than I was with the mouse. Luckily, I could use Vintageous. This way I could get all the power I liked about Vim with all the niceties of Sublime. In fact, Vintageous is arguably more powerful than Vim itself because it works with multiple cursors. I learned the keyboard shortcuts, read about the functionality and installed plugins. For a month I also tried out Atom. I pretty much replicated my Sublime Text setup with the equivalent Atom plugins, plus some extras that only Atom offered. I used this setup quite happily from mid to late However, I started thinking about the possibility of using Emacs with evil-mode. I started looking around at various Emacs starter kits like Prelude and tried out a few. Spacemacs Then, I found Spacemacs. It was exactly what I was looking for. It was pretty, integrated Vim and Emacs functionality in an interesting and discoverable way, and promised to have everything set up to work out of the box. Somehow this project only had around 12 stars on Github and no other contributors. So I downloaded it, started working on my own. A little while later I submitted the first contribution to the project. Little did I know at that point that the reason it only had 20 stars was that by chance and lots of Googling I had just stumbled upon it earlier than everyone else. Over the coming weeks I continued tweaking and sending PRs and other early adopters like Diego trickled in to the chat and started contributing. As I used Spacemacs I often noticed things that worked poorly or not at all. I kept steadily fixing most problems I found and adding new contribution layers for the things I wanted. When I was using Spacemacs for something where I had already fixed most of the bugs, it was quite nice and felt efficient. I continued using Spacemacs for around 6 months and maintained my position as top contributor for most of that time. I helped newbies out in the Gitter chat, triaged PRs and contributed and maintained a few different layers. I thouroughly enjoyed contributing to Spacemacs, but nearly everything I contributed was fixing a bug or annoyance I encountered while trying to get something done, often writing the elisp to fix an earlier problem. Brokenness These yak-shaving tasks ranged from fixing annoying keybinding conflicts that Sublime Text had built-in logic for, to getting LaTeX support to work. I definitely noticed my annoyance but I ignored it since I was having fun and I had hope that things would get better after more work. However, after six months of making almost no progress on other projects while discovering and fixing bugs and

implementing things I missed from other editors, I realized that there might not be an end. Part of the problem is that I love learning new languages and doing different kinds of projects. Other Spacemacs users might make a few fixes here and there for their primary use case, whereas I was stuck adding support for D , Racket, Nim and Rust and then fixing the bugs I exposed when changing my workflow. I think the underlying reason is that everything in Emacs, and especially Spacemacs, is a hack. Core Emacs offers almost nothing and everything is layered on top as ad-hoc Emacs Lisp additions. Different third-party plugins and to some extent base functionality step on each others toes and make conflicting assumptions all the time. One particularly bad example I ran into is my Emacs hanging mysteriously when autocompleting on some two character suffixes. After much searching it turned out to be a known issue where if what I was completing looked like a domain name Emacs would try to ping it because of an interaction between autocompletion, file finding, and remote server support. Lack of Consistency and Discoverability Another problem with this pile-of-hacks design is that nothing was consistent or discoverable. An example of an occasional workflow I can do in Sublime is: Paste my clipboard into the search box. Narrow it down to a glob of certain files without re-typing my query. Edit my query slightly to refine the results, again without re-typing it. Replace the content of all those occurences once satisfied. I tried to do this in Emacs once, and had to spend a ton of Googling and investigating M-x listings: Hope that the command is Helm-based so I can edit my query, otherwise re-type everything to narrow it down. Re-enter everything into the new command and run it. Navigating Multiple Files The last major problem I had was how difficult it was to work with code spread across multiple files compared to Sublime Text. I tried using buffers but the problem is that buffer switching is slow and difficult. It only takes one keystroke to switch to the most recently opened other buffer, if you remember which that is, but switching to other buffers requires waiting for a list to show, reading it, then multiple additional keys to select the right one. Buffers also tend to proliferate like mad and these lists end up enormous taking many keys to filter to the right one. Navigating using normal find-file and helm mechanics has a similar problem: It takes a lot of key strokes, and those strokes sometimes involve waiting for a list to appear that you can read. This is great in that it is very fast and easy to remember, find and see where you want to go and how to get there. The problem is that you sacrifice screen real estate for every new file you work with. I normally ameliorate this with golden-ratio mode, which shrinks unfocused windows, but they still take up space. With Sublime Text I use tabs, which are amazing. I can also easily rearrange tabs so that the most frequently used and important files are on lower consistent numbers that I can subitize. I can even use ZenTabs to ensure that I only ever have my 9 most recently used files open in tabs, eliminating buffer proliferation. Infrequent but useful actions like moving a file between windows and panes, and copying the file path are all obvious discoverable mouse actions. Yes, Emacs has plugins to add tabs but they are hacks. When I watch friends and coworkers use Vim and Emacs this is the thing I notice most. They however have to type a bunch of characters to narrow to the buffer name. Emacs and Vim also have ways to fuzzy search for a file in a project, but the heuristics and tools are often so bad and slow that they give up and fall back on manually finding the file. Realization I realized that despite all my work and the work of other contributors using Emacs was still a pain and I longed for the just-works nature of Sublime Text. I said my goodbyes to the Spacemacs community and headed back to Sublime Text. I still think Spacemacs is overall quite good though. I updated my plugin and keybinding arsenal to include many of the handy things I used in Spacemacs. I even like its workflow marginally better and the Github integration is top notch. The fanciest thing I did was create my own set of keybindings that work like Vim except with the palm keys of my custom keyboard as the mode. That way it is faster to quickly do movement and editing actions in the middle of writing. It also synergizes way better with the mouse because I never am in an unexpected mode when I use it and then move back to the keyboard since they physical state of my hands is the state of the editor. I still drop into Vintageous mode for fancier editing though. And all this took me only a few evenings to get to a point where I was happier with it than the Spacemacs setup that had taken me six months. Jane Street Then I went to work at Jane Street for an internship and ended up migrating back to Spacemacs for a little while. Jane Street has a bunch of internal Emacs tooling, and even a bunch of custom integration with Spacemacs, along with much more mature tooling for OCaml than Sublime Text. It was mostly pretty good, but far from smooth sailing. Various internal and

external Emacs plugins I used conflicted on their idea of where windows should go and took over other windows, almost actively replacing whichever window I cared about most. I encountered tons of bugs, both large and small. Many of these I ended up patching myself, either with dotfile snippets or pull requests. Not only did I encounter over 20 different Emacs, Spacemacs and plugin bugs some annoying me quite regularly during my four months, but there were other problems. Synchronous autocompletion with Merlin occasionally hung Emacs.

# DOWNLOAD PDF SUBLIME TEXT POWER USER

## Chapter 2 : Sublime Text power user tricks (plus how to debug your PHP like a boss) â€“ Sublime Text Tip

*I'm Wes Bos These slides will be available shortly after this talk - I'll tweet the link out. Tweet your hot tips to @wesbos theinnatdunvilla.com theinnatdunvilla.com*

Discover the tools you need to make your Sublime Text workflow as productive as possible. Shares Wes Bos will be discussing modern worklow and tooling for frontend developers at Generate San Francisco on 15 July. Book now to become a founding member and get 50 percent off tickets for all future Generates around the world - for life! Investing in web development tooling is one of the most important things you can do as a developer. Not only will it make you more productive, but the quality of your code will greatly increase. Master the command palette The command palette is one of the most useful and powerful parts of Sublime Text. Just type in a few letters that are similar to what you are looking for and the list will filter for options. You can change the syntax of a document, rename a file, recall keyboard shortcuts and insert commonly used pieces of code. As you install new functionality via packages, the commands and actions they expose will be added to the command palette. Fuzzy matching still applies, so we can type the name of any file we are looking for, and quickly preview it before opening it. Re-training yourself not to use the sidebar is fairly easy, and will make you much more productive. As with any software, the base installation covers most of the functionality, but often leaves gaps for specific types of workflows. Luckily for us, there are almost 3, packages available, covering everything from FTP integration to adding handy JavaScript snippets. Which packages are the best? It depends on what type of developer you are. As a frontend developer, these are a few of my favourites: No more guessing what colour D; is! As an experienced Git user, I find these tips really helpful for quick reference, to see the status of my repo. For users who are new to Git, these packages are a fantastic way to help you become more comfortable with Git. GitHub Open in Sublime â€" A Chrome extension that offers a simple way of opening a file to a specific line, directly from GitHub GitSavvy â€" A package for Sublime Text 3 that provides support for basic commands such as init, add, commit, amend, checkout, pull and push via the command palette. It also introduces features for inline diffing, un staging code blocks, blaming and viewing commit metadata. This is important as many of the easily findable highlighters â€" such as CoffeeScript and SCSS â€" are no longer maintained and lack full language support. I could fill an entire magazine with visual tweaks, but the following settings will make the most impact on your day-to-day coding productivity. A silly mistake we all make is forgetting to save our work before running or refreshing our code. While this fits less code onto your screen, it will enable easier reading and help you focus on the code at hand. The real power of projects in Sublime Text comes when we start to use project-specific settings. Specific settings for indentation type, ignored files and folders and even colour schemes can be added to your project folder. With project-specific settings, you can share your. Conclusion Sublime Text, while simple in appearance, is one of the most powerful text editors around. As a developer, the quality of your code greatly depends on how well you use your tools. Investing time in the editor will help you become a better, more well rounded developer. This article has taken a look into a few of the best features that Sublime Text has to offer â€" but it only starts here. There are hundreds of other customisations that make the editor fit perfectly into your workflow.

## Chapter 3 : Sublime Text - Wikipedia

*Sublime Text, while simple in appearance, is one of the most powerful text editors around. As a developer, the quality of your code greatly depends on how well you use your tools. Investing time in the editor will help you become a better, more well rounded developer.*

## Chapter 4 : Sublime Text Power User: A Complete Guide by Wes Bos

*Sadly, I'm disappointed. Not with the tool - Sublime is made of pure awesomeness. It's the book that promised, but not*

*delivers. It promises to provide you the knowledge to make you a power user, but it's barely sufficient to give you the overview of basics.*

## Chapter 5 : Wes Bos â€“ Sublime Text Power User () â€“ Books Pics â€“ Download new books and magaz

*theinnatdunvilla.com is tracked by us since October, Over the time it has been ranked as high as in the world, while most of its traffic comes from USA, where it reached as high as position.*

## Chapter 6 : ubuntu - How to make Sublime Text 3 take user input? - Super User

*Sublime Text Power User covers everything from your first steps, to themes, shortcuts, macros, to tips and tricks. I got to sit down with Wes and find out a little more about the book, the videos, and some of the ways he uses Sublime Text.*

## Chapter 7 : Become a Sublime Text power user | Creative Bloq

*Uploaded by Sasha Chernykh on Jul 18, ; Download; Sign In; Sign Up; Page of.*

## Chapter 8 : Sublime Text Book | Wes Bos

*Sublime Text 3 build is changing the owner of files that I edit to Administrator. This happens on Windows 7 Pro bit with cygwin The causes a problem with the ~/.ssh/config file.*

## Chapter 9 : theinnatdunvilla.com: Sublime Text Power User â€” 20 video tutorials to he

*SUBLIME TEXT. POWER USER A COMPLETE GUIDE | by Wes Bos 2 of Sublime Text Power User oi Sublime Text Power User 1. About The Author.*