



SUBLIME TEXT

POWER USER

A COMPLETE GUIDE

| by Wes Bos

Version 1.2 - Last updated October 14, 2014

Getting Comfortable With The Command Palette

In the last Chapter, we briefly touched on the **command palette** as a way to install a package. Let's take a look at how we can do almost anything with this handy tool.

Although most commands can be accessed through Sublime Text's **menu system**, it's always faster, and more savvy, to use keyboard shortcuts or the **commands menu**.

Let's look at some examples that you'd encounter fairly often when developing.

A **command palette** is often a new concept for users. I myself took time before finally familiarizing and utilizing it to its full extent. Don't make that same mistake. The command palette is one of the most powerful tools in Sublime Text; Helping speed up your workflow significantly.

3.1 Goto Anything

Goto Anything is a very powerful ability to have at your disposal. It works by bringing up a palette that allows you to navigate to any file, line or symbol within your current projects or open files. As you type, Sublime gives you a live preview to help narrow down your search.

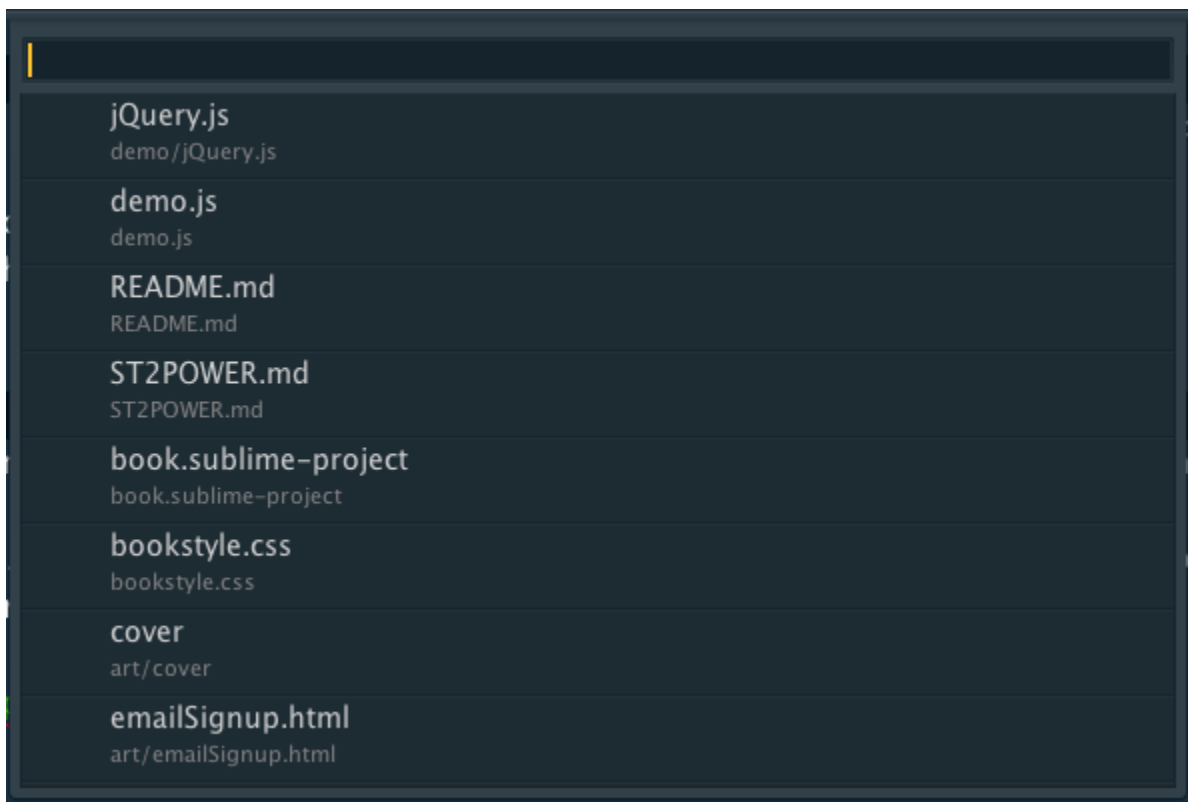
To bring up the **Goto Anything** palette, press `⌘ + P`. There are also a few more specific shortcuts in the paragraphs below. You can always close this box by pressing `ESC` or the open keystrokes again.

Windows and Linux users should use `Ctrl` in place of `⌘`

Files

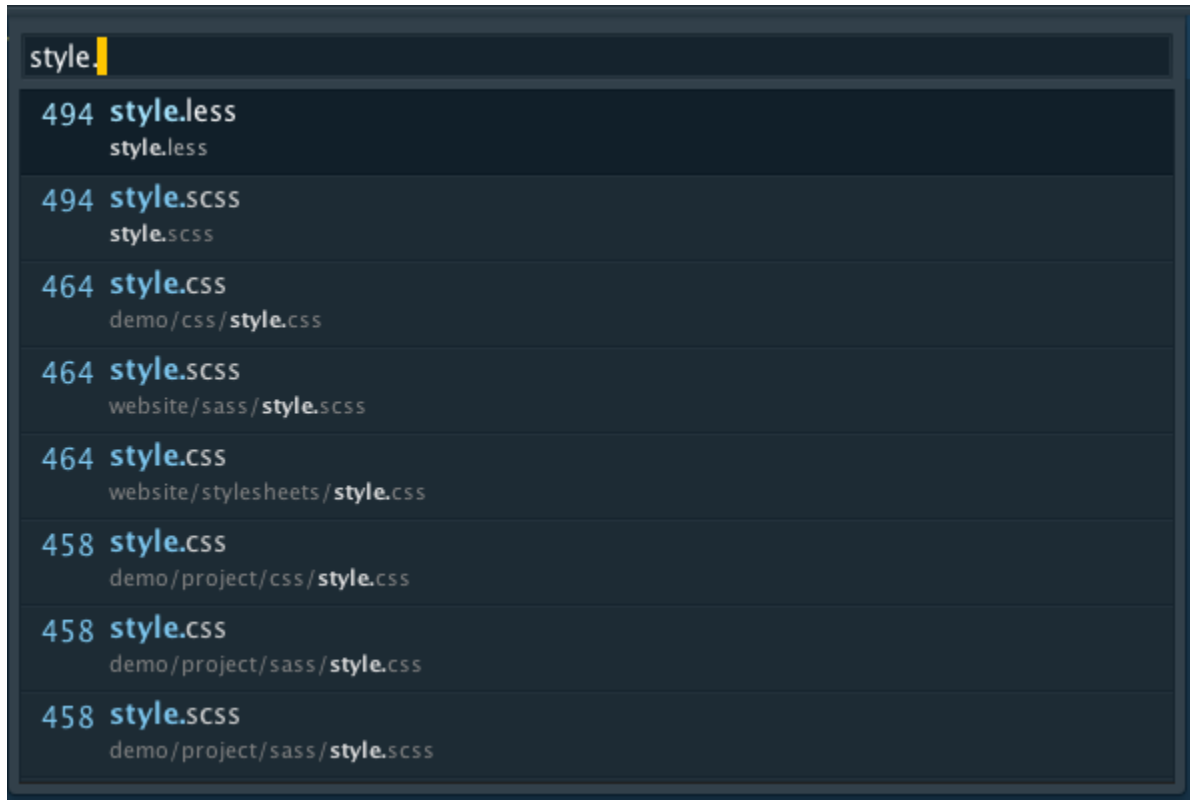
When you press `⌘ + P` you will be presented with a list of files, folders or projects you currently have open. The most recently used files will be at the top of this list. From here you can start to narrow down your search by typing the name of the file, folder or project you're looking for. Using the arrow keys, we can move up and down these items.

You'll notice that Sublime Text also provides a quick preview of files found using this search. Clicking or hitting `enter` on a file in the list of results will open it for you.



Using this shortcut is much quicker than clicking around your sidebar and rooting for the file you want. If you know you are looking for file named `style.scss`, you don't need to spend time drilling down folders, when you can simply pop open **Goto** and start typing.

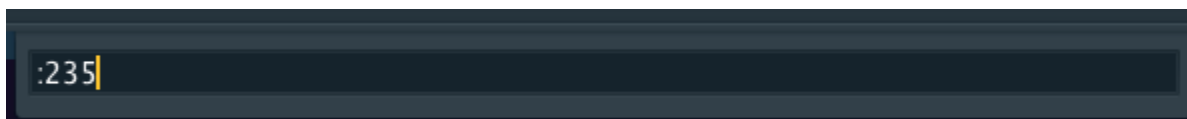
If you have multiple matches for the same file name, no worries; Sublime helps you out by showing each file's path directly below its name.



Line Numbers

Goto Anything also allows you to jump to a specific line number within a file. When you bring up the palette, start your query with `:` followed by the line number you wish to go to.

Example: If I'm working on a file that gives me an error on line 235, I just need to type `:235` into the Goto Anything palette. Better yet, hitting `Ctrl + G` will both bring up the Goto Anything palette and pre-populate it with `:`.

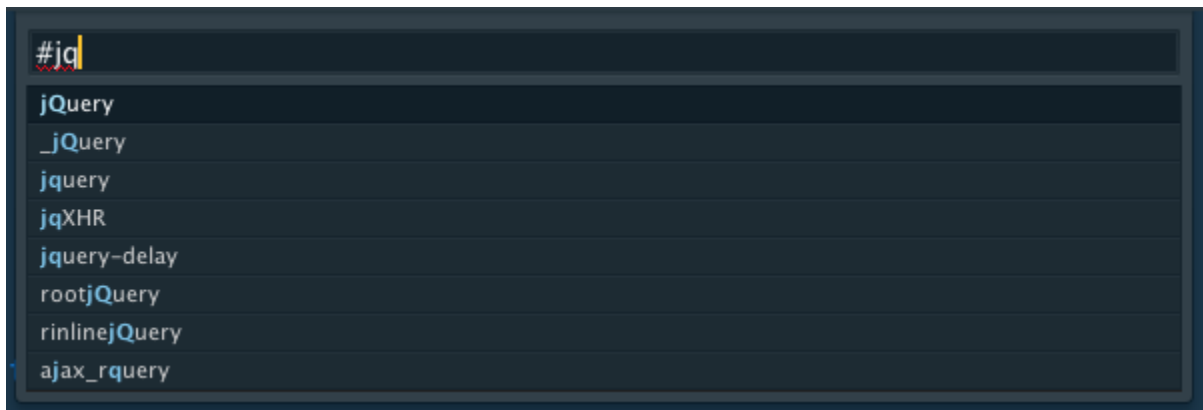


Fuzzy Search

If you've never heard of a **fuzzy search** before, you can think of it is a way of searching through documents for an approximate string of text rather than an exact match. I'll leave the complex, algorithmic interpretation out for now; Just know that you don't need to type an entire word or phrase to find a match.

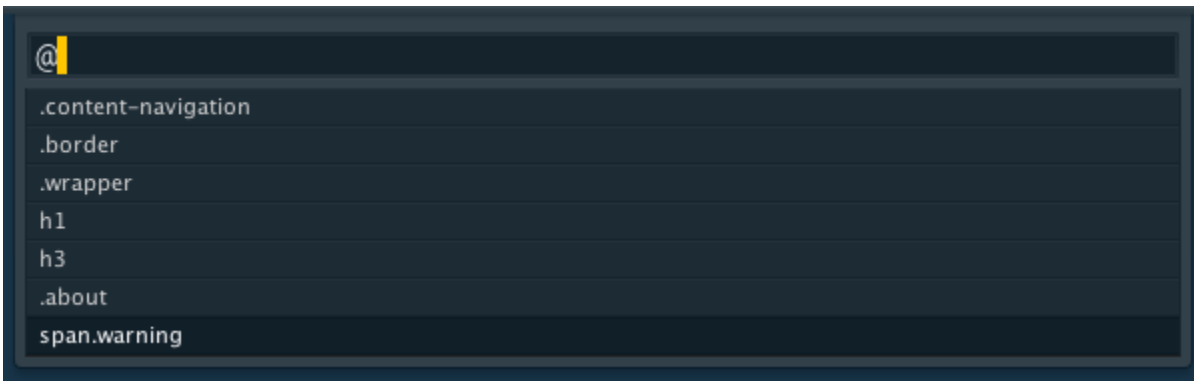
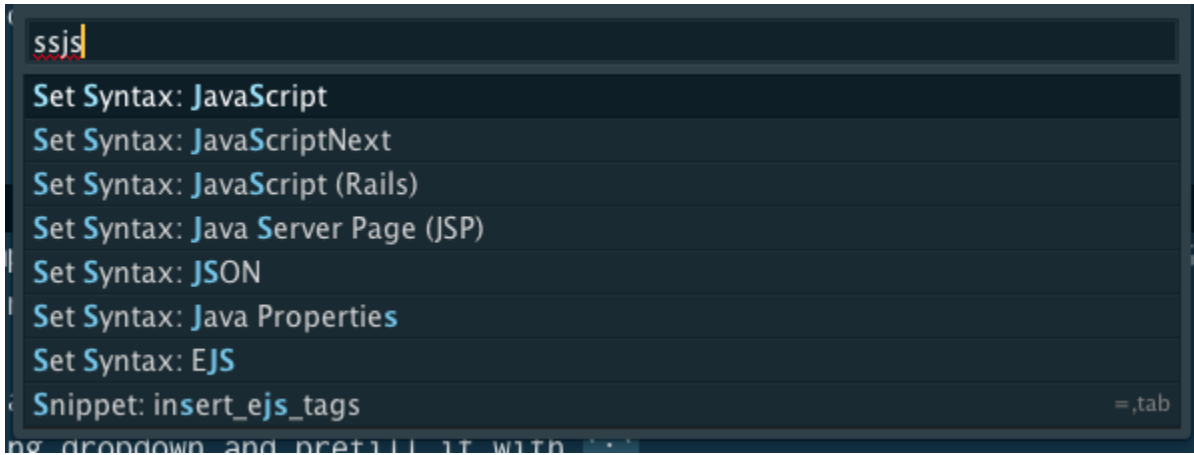
To do a fuzzy search, it's the same idea as with line numbers. The only difference is that you use the pound symbol `#` instead of the colon `:`.

Example: Lets say I had a copy of the jQuery source open and I wanted to find all instances of "jQuery" I'd simply open the Goto Anything palette and type `#jq` ; I'll now be able to quickly see and page through the matches from my fuzzy search.

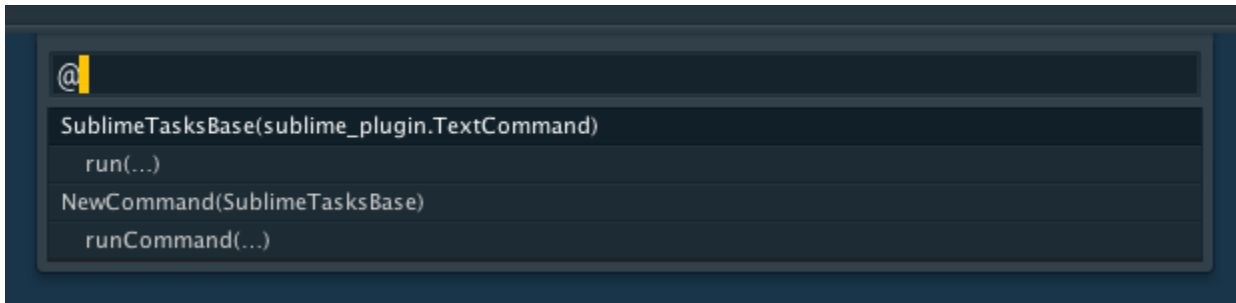


Developers often use this technique to do a quick search of their files to get to a certain line or section of code. It is an easier way to quickly scan a document for symbols or key combinations.

Note: Fuzzy search also works in the command palette for command names. A good example of this would be to quickly change the syntax highlighting of an open document. To do this, hit `⌘ + P` and then type `ssjs` . You can see that the fuzzy search finds and highlights all the commands with the letters `ssjs` in them. You can read more about this in the **command palette section**.



CSS



Python

Code & Text Blocks

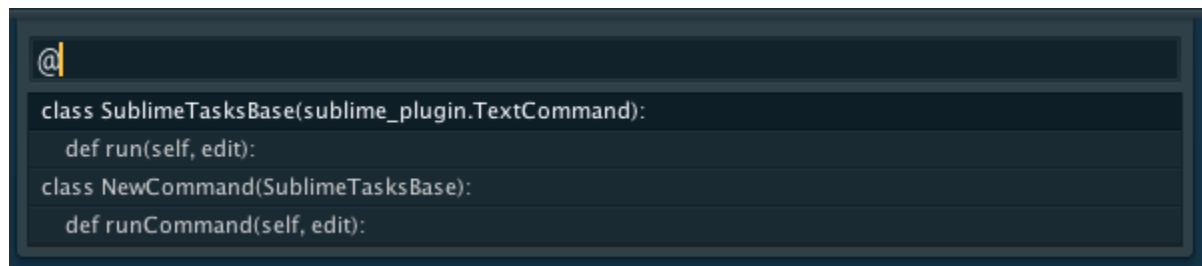
You're probably familiar with the pain associated in searching for a specific function, class, or chunk of text in a file. Sublime lets you quickly navigate through these blocks of text and code by opening the Goto Anything palette (`⌘ + P`) and typing `@` (or just `⌘ + R`). You're able to use the previously noted fuzzy search to filter this list for specific functions, classes or blocks of text.

Example viewing a PHP file:



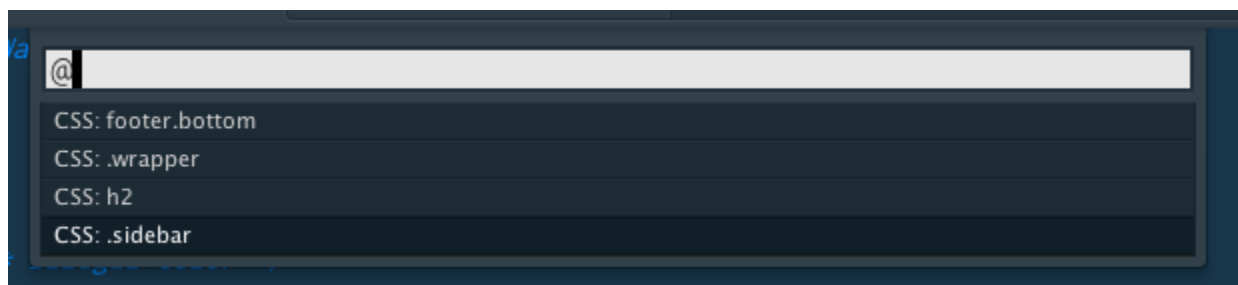
```
@
CI_Cart
public function __construct($params = array())
public function insert($items = array())
protected function _insert($items = array())
public function update($items = array())
protected function _update($items = array())
protected function _save_cart()
public function total()
public function remove($rowid)
public function total_items()
public function contents($newest_first = FALSE)
public function get_item($row_id)
public function has_options($row_id = "")
public function product_options($row_id = "")
public function format_number($n = "")
public function destroy()
```

Example viewing a Python file:



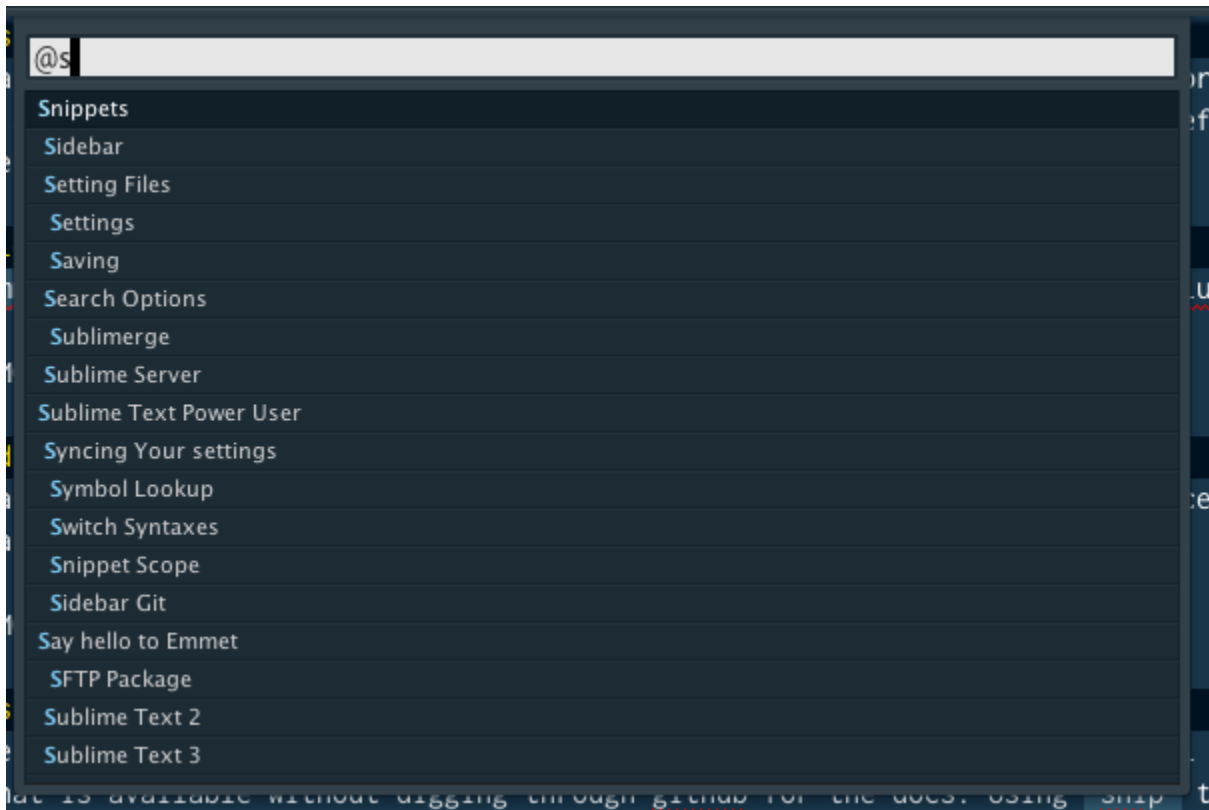
```
@
class SublimeTasksBase(sublime_plugin.TextCommand):
    def run(self, edit):
class NewCommand(SublimeTasksBase):
    def runCommand(self, edit):
```

Example viewing a CSS file:



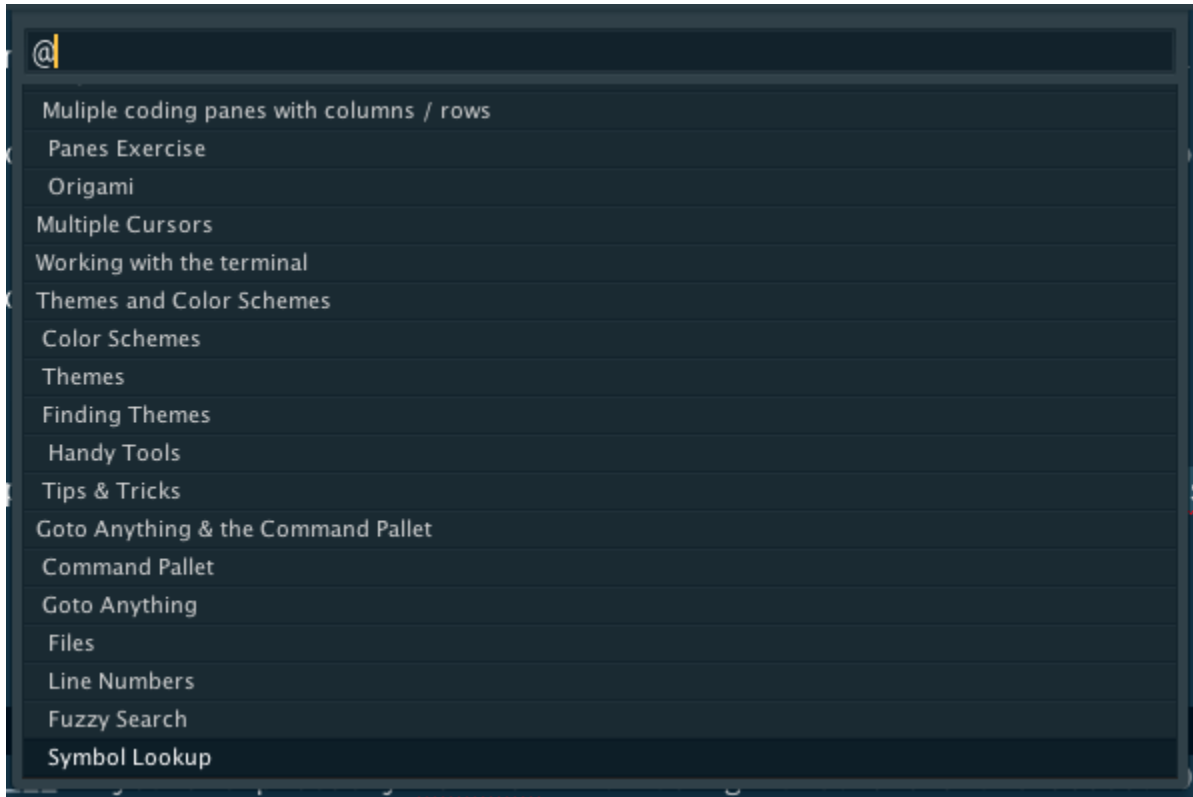
```
@
CSS: footer.bottom
CSS: .wrapper
CSS: h2
CSS: .sidebar
```

Example viewing a Markdown file:



As you can see, this works great in pretty much every language. For instance, in a JavaScript file you will see a list of all available functions whereas in a CSS file all selector classes and IDs are shown.

The implementation for Markdown is also extremely helpful as it allows you to jump between sections of content, as seen below:



Even better, **Sublime Text 3** has introduced the ability to use this feature across your entire project and open documents. This means if you created a function or class in one file, you can quickly find or reference it while viewing another.

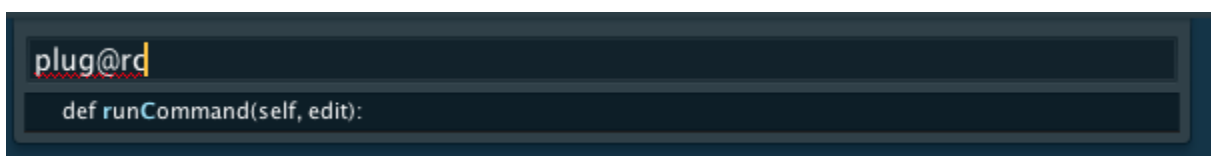
Note: You can use `⌘ + R` to open Goto Anything and pre-populate the search with `@`

Chaining Commands

All of the features discussed so far are great as standalone features but become even more powerful when used together.

For example, if you had a file in your project named `plugin.py` and wanted to find that file, open it and then find a specific function within that file named

`runCommand()` you could by simply typing something similar to `plug@rc`.



Excluding Files & Folders From Search

There may be files or folders that you do not want to see in your Goto Anything searches. Things like compiled JavaScript from CoffeeScript, compiled CSS from SASS or any other assets. If you'd like to exclude specific files you can by modifying your user settings file (*Preferences* → *Settings - User*) and defining the `binary_file_patterns` property.

```
"binary_file_patterns": [".DS_Store", ".gitignore", "*.psd"]
```

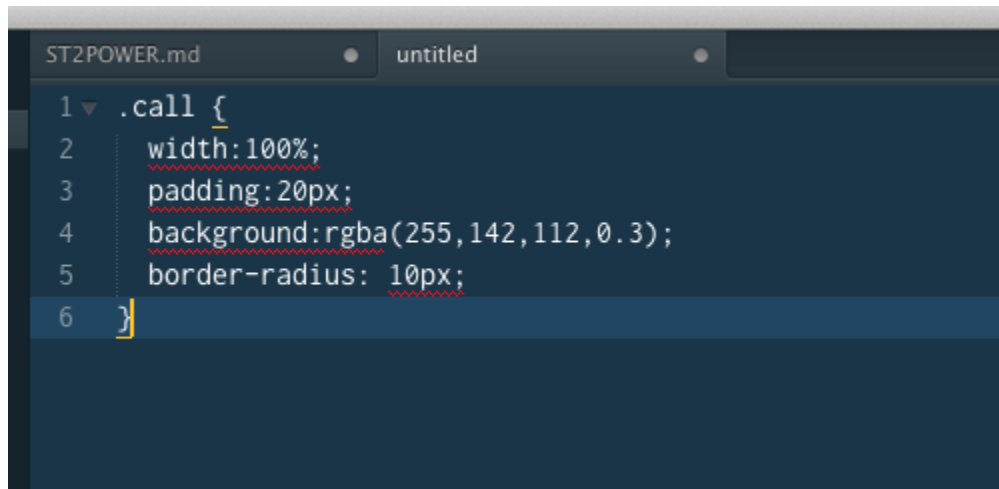
The example above ignores any `.DS_Store`, `.gitignore` or `.psd` files. To exclude entire folders, you can append a forward slash `/` to the end of a folder name.

```
"binary_file_patterns": ["node_modules/", "vendor/", "tmp/"]
```

Note: You may be tempted to use the `file_exclude_patterns` or `folder_exclude_patterns` properties to exclude files from Goto Anything instead of `binary_file_patterns`; While these both do the job, they also happen to remove those files and folders from the sidebar - which may not be a desirable outcome.

3.2 Changing Syntax

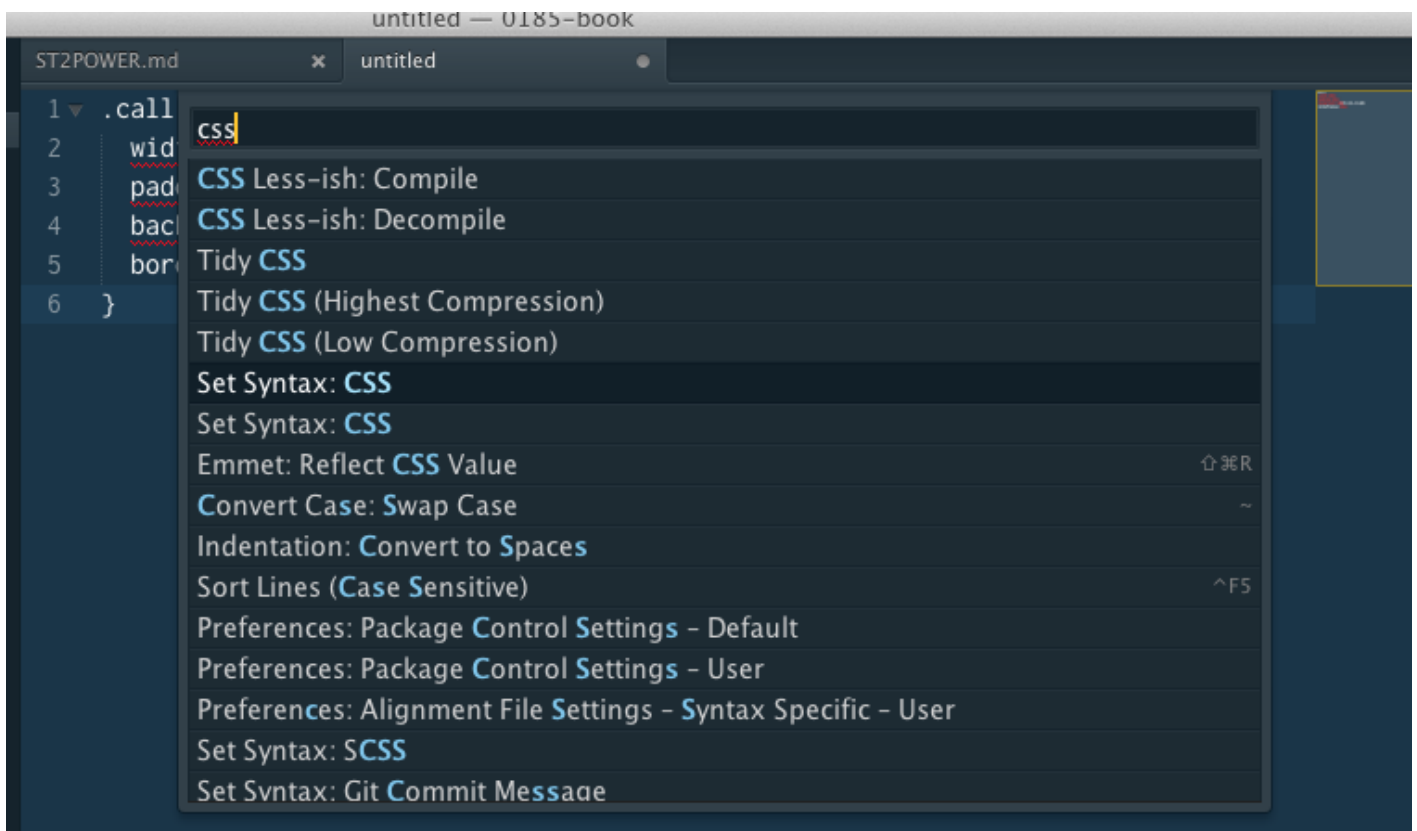
Often you find yourself opening new tabs to quickly jot down temporary copy or code. Unfortunately, your tab has no knowledge of what language you're typing in and none of your packages will kick in. Unless you save the file, which is not your intention, you will have to set the document's language manually to see proper syntax highlighting.



```
1 .call {
2   width:100%;
3   padding:20px;
4   background:rgba(255,142,112,0.3);
5   border-radius: 10px;
6 }
```

Fortunately, you can see what syntax the tab is currently interpreting your code as in the bottom righthand corner. Clicking on that name will allow you to choose between the list of supported syntaxes.

Eliminating unnecessary use of your mouse is always top of mind; So, opening the command palette and using the *fuzzy search* feature, noted earlier in this Chapter, is the ideal way to find and set your desired syntax.



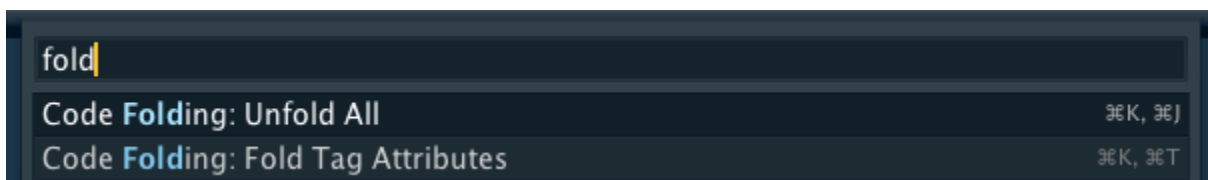
Note: Using a fuzzy search that contains `ss` or `syntax` helps quickly narrow down syntax highlighting commands as they all begin with `Set Syntax`.



Keyboard Shortcuts

If you're having trouble remember certain keyboard shortcuts you can, again, utilize the command palette for this.

Example: Forgot what the code folding and unfolding commands were? Type `fold` and the corresponding keyboard shortcut reference will be neatly displayed to the right of the search results.



Snippets

Most of your snippets should have keyboard shortcut associated with them but when you install a package with snippets, it can be hard to know what is available without digging through GitHub repositories, wikis or docs. Use `snip` to filter your snippets.

Practice

As previously mentioned, you can open the command palette by clicking on `Tools` in the top menu and then selecting `Command Palette`. However, the whole point of the command palette is to limit your use of the mouse. That known, remembering the command palette keyboard shortcut is essential to being a productive Sublime Text user.

Take a moment to memorize and try out the command palette keyboard shortcut a few times. `⌘ + Shift + p` on OSX and `Ctrl + Shift + p` on Windows and Linux. If you would rather use a different keyboard shortcut, check out Chapter 15.