Sublime Text 3 官方文档(中文)

书栈(BookStack.CN)

目　录

致谢

当前文档 《Sublime Text 3 官方文档(中文)》 由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-06-17。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常工作、生活和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址：http://www.bookstack.cn/books/Sublime-Text-3-Documentation

书栈官网：http://www.bookstack.cn

书栈开源：https://github.com/TruthHun

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

使用

列选择

列选择

概述

列选择可用于选择一个文件里的矩形区域。列选择并不是通过一个单独模式完成的，而是使用多重选择。

你可以添加选区域来选择多个文本块，或者减少选区来移除选择块。

使用鼠标

不同的平台要使用不同的鼠标按钮：

OS X

- 鼠标左键 + Option
- 或：鼠标中键
- 添加到选区：Command
- 从选区移除：Command+Shift

Windows

- 鼠标右键 + Shift
- 或：鼠标中键
- 添加到选区：Ctrl
- 从选区移除：Alt

Linux

- 鼠标右键 + Shift
- 添加到选区：Ctrl
- 从选区移除：Alt

使用键盘

OS X

- Ctrl + Shift + Up

- Ctrl + Shift + Down

Windows

- Ctrl + Alt + Up
- Ctrl + Alt + Down

Linux

- Ctrl + Alt + Up
- Ctrl + Alt + Down

原文：

> *http://feliving.github.io/Sublime-Text-3-Documentation/column_selection.html*

键盘多区域选择

键盘多区域选择

添加一行

添加上面一行或者下面一行到选区，可以使用Ctl+Alt+Up 和 Ctrl+Alt+down (OS X: Ctrl+Shift+Up 和 Ctrl+Shift+Down).

如果你选择过多，可以撤销选择 (Ctrl+U, 或者 Command+U on OS X) 进行后退。

把选区分割成每一行

选中一个多行的区块，然后把每一行分成一个选区，可以使用Ctrl+Shift+L, 或 Command+Shift+L on OS X.

快速添加下一个

添加下一个当前发现的单词到选区，Windows和Linux下使用Ctrl+D ，或者OS下使用Command+D。如果当前没有选中的单词，第一次则选中当前光标所在的单词，重复使用将选择下一个相同的单词。

同样，如果选择过多，可以使用(Ctrl+U, 或者 Command+U on OS X) 进行撤销，后退一步。

使用快速选择下一个时，个别单词可以跳过，Windows 和 Linux下快捷键是Ctrl+K,Ctrl+D, 或者OS X下Command+K,Command+D。

查找所有

添加所有的发现的当前单词到选区，可使用查找所有: Alt+F3 on Windows and Linux, 或者Ctrl+Command+G on OS X.

单一选区

从多选区切换到单一选区，使用Escape.

原文:

> *http://feliving.github.io/Sublime-Text-3-Documentation/multiple_selection_with_the_keyboard.html*

自动完成

自动完成

概述

自动完成会根据你的输入显示补全列表，所以你可以通过输入简短的字母来完成填入长的单词。 默认编辑源代码和HTML(输入<字符之后启动提示)时是开启的。

禁用自动完成

自动完成可以通过auto_complete来配置。把下面这行加入到Preferences/File Settings - User里面可以禁用:

```
1. "auto_complete": false
```

如果自动完成被禁用，补全列表弹出层可以手动触发，或者通过tab键来完成输入最接近的自动补全匹配，而不显示补全列表提示。

手动显示补全列表

如果未显示可以通过 Ctrl+Space 显示提示列表。如果已经显示，可以选择下一项。

通过Tab键完成

默认情况下，补全提示列表中选中的项会在回车之后自动填入。这个可能会导致歧义，是需要换行还是自动完成补全。通过设置auto_complete_commit_on_tab 为true， 回车将会插入新行，tab键将自动完成补全。这样做的另一个好处就是: 让Sublime Text知道没有模糊的定义， 它会把你可能想要的最为匹配的项展现在第一个位置。

建议启用Tab键完成自动补全，不过可能需要一点时间适应这个习惯。

原文:

> *http://feliving.github.io/Sublime-Text-3-Documentation/auto_complete.html*

Tab完成

Tab完成

概述

Tab完成可以通过Tab键快速完成单词的输入。启用时，按tab键将会把光标左边文本扩展为最为匹配的词，使用了Sublime Text的模糊匹配算法。

默认情况下Tab完成是启用的。

禁用Tab完成

有些时候Tab完成并不令人满意。把下面这行配置添加到Preferences/File Settings - User可以禁用它:

```
1. "tab_completion": false
```

插入Tab字符

插入Tab字符而不是触发自动补全，可以使用shift+tab.

更换补全内容

有些时候自动补全的文本并不是我们想要的，可以通过ctrl+space来更换候选词，这样就会回退补全的内容，并显示标准的自动完成列表。

另外，可以再次按tab键轮询下一个补全内容

自动完成的来源

自动完成的信息来源，或者要创建自己的自动完成信息可以参考sublimetext.info.

原文:

> *http://feliving.github.io/Sublime-Text-3-Documentation/tab_completion.html*

免注意力分散模式

免注意力分散模式

概述

换句话说就是注意力集中模式，这种模式下代码编辑区域将变成全屏。所有UI部件都会被影藏，但是仍然可以访问。可以通过View/Enter Distraction Free Mode 菜单进入此模式。

这种模式下，所有UI部件(侧边栏，预览地图，状态栏等等)将影藏。可以通过View菜单里选择性的显示某个部分，这些配置同样将会在下一次进入该模式时有效。

定制

该模式下可以应用某些配置。默认配置(在Packages/Default/Distraction Free.sublime-settings):

```
{
    "line_numbers": false,
    "gutter": false,
    "draw_centered": true,
    "wrap_width": 80,
    "word_wrap": true,
    "scroll_past_end": true
}
```

可以通过编辑Packages/User/Distraction Free.sublime-settings文件来自定义这些配置，可通过Preferences/File Settings - More菜单访问到。

上面wrap_width设置没太大意义。80表示代码编辑区域边界位置在第80个字符位置。你可以设置成更大的值，或者设置为0表示整个窗口的宽度。

原文:

> *http://feliving.github.io/Sublime-Text-3-Documentation/distraction_free.html*

Vintage模式

Vintage模式

概述

Vintage是Sublime Text的vi模式编辑包。 可以使用组合vi命令来调用Sublime Text的功能，包括多重选择。

Vintage模式是开放开发的，而且非常欢迎进行完善和扩展。如果你愿意做点贡献的话，可以在GitHub页面上找到更多细节。

启用Vintage

Vintage默认是禁用的， 通过ignored_packages 配置。如果要从ignored packages列表中移除"Vintage"的话可以通过下面的方式编辑：

- 选择Preferences/Settings - Default菜单
- 编辑ignored_packages配置， 修改：

```
1.     "ignored_packages": ["Vintage"]
```

成：

```
1.     "ignored_packages": []
```

然后保存文件。

- Vintage模式则已启用—你可以看到"INSERT MODE"显示在状态栏了。

Vintage默认是插入模式。可以添加：

```
1.     "vintage_start_in_command_mode": true
```

这项配置到User Settings里。 ## 包含哪些功能 Vintage包含大部分的基本命令：d (delete), y (copy), c (change), gu (lower case), gU (upper case), g~ (swap case), g? (rot13), < (unindent), and > (indent)。 同时也包含许多移动操作，包括l, h, j, k, W, w, e, E, b, B, alt+w (move by sub-words), alt+W (move backwards by sub-words), - , ^, %, 0, G, gg, f, F, t, T, ^f, ^b, H, M, and L. 文本对象的支持，包括词，引号，括号和标签。 重复点号('.')也是支持的， 用于重复指定次数的命令和移动。寄存器也是支持的，有宏命令和书签。许多其它混合命令也支持，比如*, /, n, N, s, S 等等。 ## 哪些没有 插入模式时常规的Sublime Text编辑模式，绑定的是Sublime Text常用的快捷键：并没有模仿vi 插入模式的键盘绑定。 通过via命令面板执行的Ex commands没有实现，除了:w和:e。 ## Under the Hood Vintage模式完全是参考via键盘绑定和基于插件API来实现的—你可以自由的浏览Vintage包，看看它们是如何组合在一起的。举个例子，如果你要把 "jj" 绑定为退出插入模式，你可以像下面这样

天剑键盘绑定：

```
{ "keys": ["j", "j"], "command": "exit_insert_mode",
    "context":
    [
        { "key": "setting.command_mode", "operand": false },
        { "key": "setting.is_widget", "operand": false }
    ]
}
```

OS X Lion

Lion系统下，按住某个键不会重复，但是会弹出一个菜单选择字符变更。在命令模式下这样并不是很好，如果你想禁用它，可以通过在终端里输入下面这个语句：

```
defaults write com.sublimetext.2 ApplePressAndHoldEnabled -bool false
```

Ctrl Keys

Vintage支持下面这些ctrl key的绑定：

- Ctrl+[: Escape
- Ctrl+R: Redo
- Ctrl+Y: Scroll down one line
- Ctrl+E: Scroll up one line
- Ctrl+F: Page Down
- Ctrl+B: Page Up不过，因为这些会与Sublime Text的其它键盘绑定冲突，Windows和Linux下默认是禁用的。可以通过vintage_ctrl_keys来配置：

  ```
      "vintage_ctrl_keys": true
  ```

Ex Mode

可以参考VintageEx这里查看Vintage的Ex mode。

原文：

> *http://feliving.github.io/Sublime-Text-3-Documentation/vintage.html*

Projects

Projects(项目)

概述

Projects在Sublime Text中由2个文件组成：sublime-project文件，包含项目的定义，和sublime-workspace文件，包含用户特定数据，比如打开的文件和每个文件的修改。

按照一般原则，sublime-project需要纳入版本控制，而sublime-workspace文件不需要。

Project Format

sublime-project文件是JSON格式，支持三个一级配置节点：folders，包含引入的文件目录，settings，包含需要重写的file-setting，以及build_systems，指定项目的构建系统。下面是一个例子：

```
{
    "folders":
    [
        {
            "path": "src",
            "folder_exclude_patterns": ["backup"],
            "follow_symlinks": true
        },
        {
            "path": "docs",
            "name": "Documentation",
            "file_exclude_patterns": ["*.css"]
        }
    ],
    "settings":
    {
        "tab_size": 8
    },
    "build_systems":
    [
        {
            "name": "List",
            "shell_cmd": "ls -l"
        }
    ]
}
```

Folders

每个文件夹都需要有一个path(路径), 和其它可选配置file_exclude_patterns, file_include_patterns, folder_exclude_patterns, folder_include_patterns and follow_symlinks。路径是相对于project目录的位置，或者完整绝对路径。Folders也可以指定一个name配置，用于显示在侧边栏里。

如果从老版本的项目转换过来可能会在folders下有一个mount_points。 如果还想使用这个移除的模式，需要转换成上面这种格式。

Settings

配置(Settings)可以在这回使用settings这个key值来指定， 可以覆盖用户配置规则(user settings)。注意它不会覆盖特定的语法配置(syntax specific settings)。

构建系统(Build Systems)

Build Systems指定了一组内联的构建系统定义. 除了一般的构建系统的配置，还需要为每个构建系统指定一个name。可用的构建系统列举在Tools/Build Systems菜单下。

原文:

> *http://feliving.github.io/Sublime-Text-3-Documentation/projects.html*

自定义

配置

配置

概述

Sublime Text有很多不同的配置来自定它的行为。配置可以通过编辑文本文件来修改：虽然这比使用GUI有点麻烦，不过这会是一个更灵活的系统。

Settings

查看所有可用的配置以及每个配置的描述，可以查看Packages/Default/Preferences.sublime-settings。可以通过Preferences/Settings - Default菜单来访问这个文件。

如果你发现你想要修改的配置，把它们添加到用户配置(User Settings)里面 (可通过Preferences/Settings - User菜单访问)，这样即便Sublime升级也会被保存。

Settings文件

配置文件的加载和启用是按下面这个顺序：

- Packages/Default/Preferences.sublime-settings
- Packages/Default/Preferences ().sublime-settings
- **Packages/User/Preferences.sublime-settings**
-
- Packages//.sublime-settings
- **Packages/User/.sublime-settings**
-

通常情况下，你应该把你的配置放在Packages/User/Preferences.sublime-settings里。如果你要给特定的文件类型指定配置，比如，Python, 应该放在Packages/User/Python.sublime-settings文件中。

配置文件示例

试着把这些保存为Packages/User/Preferences.sublime-settings

```
{
    "tab_size": 4,
    "translate_tabs_to_spaces": false
}
```

单独语法配置

在每一种语法配置的基础上可以指定一些配置。通常用于根据文件类型指定不同语法高亮颜色方案。

可以通过Preferences/Settings - More/Syntax Specific - User菜单来编辑当前文件的语法配置。

单独的项目配置

可以在project配置的基础上为每个项目进行单独的配置。可以参考Project相关的文档

免注意力分散(Distraction Free)的配置

Distraction Free模式有一个额外的配置文件(Distraction Free.sublime-settings)。可以把Distraction Free模式相关的配置放在这里，通过Preferences/Settings - More/Distraction Free - User菜单来访问。

修改键盘绑定的配置

toggle_setting命令可用于触发设置。例如，将一个快捷键绑定为触发word_wrap，you can use (in Preferences/Key Bindings - User):

```
{
    "keys": ["alt+w"],
    "command": "toggle_setting",
    "args":
    {
        "setting": "word_wrap"
    }
}
```

set_setting命令可用于把一项setting设置为指定的value。比如，下面这项键盘绑定把当前文件的语法方案设置为Cobalt主题:

```
{
    "keys": ["ctrl+k", "ctrl+c"],
    "command": "set_setting",
    "args":
    {
        "setting": "color_scheme",
        "value": "Packages/Color Scheme - Default/Cobalt.tmTheme"
    }
}
```

上面的这些配置修改是缓冲区的特定配置：它们会覆盖任何地方保存在配置文件里的配置，但是只对当前文件有效。

故障排除

由于配置可以放在多个不同的地方，如果要查看当前文件应用的某项配置，可以在控制台通过下面的语句来查看:

```
view.settings().get('font_face')
```

原文：

> *http://feliving.github.io/Sublime-Text-3-Documentation/settings.html*

修改字体

字体设置

概述

通过菜单Preferences/Settings - User,添加下面这行配置就可以修改字体:

```
1.      "font_face": "Courier New",
        "font_size": 10
```

保存则立即会生效。
可以通过Preferences/Settings - Default菜单来查看其它的配置。在Sublime Text中，这里有很多可自定义的东西。

原文:

> *http://feliving.github.io/Sublime-Text-3-Documentation/font.html*

缩进

缩进设置

概述

缩进设置决定了tab符缩进的大小，控制tab键是插入tab符号还是空格。除了自动检测之外，它们可以自定义为全局，某种文件类型，或者某个文件。

设置

|tab_size|数字。插入的空格数
|translate_tabs_to_spaces|Boolean, 如果为true，按tab键将会输入空格替代，而不是tab字符。
|detect_indentation|Boolean, 默认为true, tab_size和translate_tabs_to_spaces将会在文件载入是自动计算。
|use_tab_stops|Boolean, 如果translate_tabs_to_spaces为true, use_tab_stops将会使tab和backspace在下一个tab停止时insert/delete

配置文件

配置文件将会按下面这个顺序应用:

- Packages/Default/Preferences.sublime-settings
- Packages/Default/Preferences ().sublime-settings
- **Packages/User/Preferences.sublime-settings**
- Packages//.sublime-settings
- **Packages/User/.sublime-settings**

通常情况下，你应该把你的配置放在Packages/User/Preferences.sublime-settings里。如果你要给特定的文件类型指定配置，比如，Python, 应该放在Packages/User/Python.sublime-settings文件中。

配置文件示例

试着把这些保存为Packages/User/Preferences.sublime-settings

```
{
    "tab_size": 4,
    "translate_tabs_to_spaces": false
}
```

单独语法配置

可以在基础配置之上指定单独的语法配置。在Preferences/Settings - More/Syntax Specific - User菜单下。

缩进的检测

当一个文件载入时，它的内容会被检查，tab_size和translate_tabs_to_spaces设置将会应用到该文件。状态栏将会报告发了什么。尽管编辑器会处理的很好，如果想要把它禁用的话，可以通过detect_indentation来设置。

缩进检测可以手动执行，通过View/Indentation/Guess Settings From Buffer菜单执行detect_indentation命令。

Tab和空格之间转换

View/Indentation菜单里有命令可以将当前文件中的空白在tab符和空格符之间转换。这几个菜单项执行的是expand_tabs和unexpand_tabs命令。

自动缩进

自动缩进猜测会在换行时给每一行添加一定数量空白符。由下面这个配置控制：

|auto_indent|Boolean, 默认是开启。
|smart_indent|Boolean, 默认是开启。具有一点小聪明的自动缩进，比如，在一个if语法片段的下一行进行缩进。
|trim_automatic_white_space|Boolean, 默认开启。当断行时由auto_indent去除行头尾的空白。
|indent_to_bracket|Boolean, 默认禁用。缩进时根据第一个前括号来空白数。像下面这样：

```
1. use_indent_to_bracket(to_indent,
2.                       like_this);
```

原文：

> *http://feliving.github.io/Sublime-Text-3-Documentation/indentation.html*

拼写检查

拼写检查

概述

Sublime Text使用了Hunspell来做拼写检查的支持。可以从OpenOffice.org Extension List获取额外的词典。

Sublime Text可使用的词典在这里https://github.com/SublimeText/Dictionaries

词典

Sublime Text目前只支持UTF-8编码的词典。大部分词典并不是UTF-8编码的，而是使用一种更紧凑的编码。所以在Sublime Text中使用词典，需要将它转换成UTF-8编码。

如果你有一个UTF-8编码的词典，可以将它放在一package中，比如，Packages/User， 可以通过Preferences/Browse Packages菜单访问。然后就可以在View/Dictionary菜单中选择词典了。

配置

有两项配置会影响拼写检查：spell_check， 控制是否启用，dictionary， 指定词典的路径。示例：

```
1. "spell_check": true,
2. "dictionary": "Packages/Language - English/en_US.dic"
```

命令

- next_misspelling: 选择下一个拼写错误
- prev_misspelling: 选择上一个拼写错误
- ignore_word: 指定word单词添加到忽略列表中。

原文：

> *http://feliving.github.io/Sublime-Text-3-Documentation/spell_checking.html*

包(Packages)

Contents

Packages

概述

Packages是Sublime Text的一组资源文件集合，用于：插件，语法高亮定义，菜单，代码片段等等。Sublime Text只附带几个包，更多的是用户创建的。

Packages以.sublime-package文件格式保存，其实是zip格式，只不过用了另外一个后缀名称。Packages也可以以未压缩的目录保存，或者2种方式混合：包目录下的文件将会覆盖.sublime-package格式压缩包中的文件。

目录位置

zip压缩的包放在：

- /Packages
- /Installed Packages
  目录包可以放在：
- /Packages
  例如，Python包放在<executable_path>/Packages/Python.sublime-package， 任何放在<data_path>/Packages/Python目录下的文件将会覆盖.sublime-package压缩包中的文件。

一般情况下，<executable_path>/Packages下放Sublime Text的自带包，<data_path>/Installed Packages下放第三方用户开发的包。

特殊包

有2个特殊包：Default和User。Default优先顺序总是排在第一，User优先顺序总是排在最后。包的优先顺序会影响包之间的文件合并，比如Main.sublime-menu。任何包都可能会包含一个Main.sublime-menu文件，但是却不会覆盖掉主菜单，而是根据包顺序把这个文件进行合并。

Default和User之外的包按照字母顺序排序。

创建一个新的包

要创建一个新的包，只需要简单的在<data*path>/Installed Packages*目录下新建一个目录就可以了。可以通过_Preferences/Browse Packages*菜单打开这个目录。

覆盖Zip包中的文件

在Packages/<Package Name>目录下建立一个相同名字的文件即可。

例如要覆盖Sublime Text自带的Python.sublime-package包中的function.sublime-snippet文件，在<data_path>/Packages下创建Python目录，然后把自己的function.sublime-snippet文件放在改目录下即可。

原文：

> *http://feliving.github.io/Sublime-Text-3-Documentation/packages.html*

其它

恢复到新安装状态

恢复到新安装状态

概述

Sublime Text可以通过移除data文件夹来恢复到初始安装状态。根据操作系统，这个文件夹的位置为：

- OS X: ~/Library/Application Support/Sublime Text 3
- Windows: %APPDATA%\Sublime Text 3
- Linux: ~/.config/sublime-text-3
  按下面操作恢复到初始安装状态：
  i. 退出Sublime Text
  ii. 删除上面列举的数据目录
  iii. 启动Sublime Text

  当重新启动是，将会创建一个新的data文件夹，跟第一次运行Sublime Text时一样。注意，这也会移除你所有的settings和packages。

OS X Lion

在Lion系统里，~/Library文件夹默认是影藏的。可以在Finder里通过Go/Go to Folder菜单输入 ~/Library 来浏览。

Windows

在Windows下，缓存文件保存在单独的位置，%LOCALAPPDATA%\Sublime Text 3，使用漫游数据文件(roaming profiles)来提高性能。

原文：

> *http://feliving.github.io/Sublime-Text-3-Documentation/revert.html*

OS X命令行

OS X命令行

概述

Sublime Text包含一个命令行工具,subl, 通过命令行处理管理。可以在Sublime Text中打开文件和项目，也可以作为一个unix下的EDITOR工具，例如git和subversion。

安装

首先给subl做一个链接。假设你把Sublime Text放在Applications目录下，并且在~/bin目录已经在path中，你可以运行：

```
ln -s "/Applications/Sublime Text.app/Contents/SharedSupport/bin/subl" ~/bin/subl
```

用法

运行 subl –help,

```
Usage: subl [arguments] [files]         编辑指定的文件edit the given files
   or: subl [arguments] [directories]   打开指定的目录
   or: subl [arguments] -               编辑stdin

Arguments:
  --project <project>: 载入指定的project
  --command <command>: 运行指定的命令
  -n or --new-window:  打开一个新的窗口
  -a or --add:         添加文件夹到当前窗口
  -w or --wait:        返回前等待文件关闭
  -b or --background:  不激活该应用程序
  -s or --stay:        文件关闭后保持应用程序激活状态
  -h or --help:        显示帮助并退出
  -v or --version:     显示版本信息并退出

如果从标准输入--wait是隐式的。 使用--stay当文件关闭是不切换到后台控制台(只与是否有等待的文件有关)。

文件名可以通过加:line或者:line:column后缀来指定打开的定位。
```

EDITOR

使用Sublime Text作为许多命令提示输入的编辑器，可以设置EDITOR环境变量为：

```
1. export EDITOR='subl -w'
```

特别是-w会指定subl控制器不会退出，直到文件关闭。

原文：

> *http://feliving.github.io/Sublime-Text-3-Documentation/osx_command_line.html*

API

API手册

API手册

Sublime API

- View
- Selection
- Region
- Edit
- Window

- Settings

基类

- EventListener

- ApplicationCommand
- WindowCommand
- TextCommand

插件示例

在Default包里有下面这些内置的插件可以作为参考看看：

- Packages/Default/delete_word.py 删除光标左边或者右边的一个单词
- Packages/Default/duplicate_line.py复制当前行
- Packages/Default/goto_line.py提示用户输入，然后更新选择点
- Packages/Default/font.py示了如何使用settings
- Packages/Default/mark.py 用了add_regions() 往行头槽里插入图标
- Packages/Default/trim_trailing_whitespace.py保存前修改缓冲区

插件生命周期

在导入期间，插件不能调用API方法，除了可以调用sublime.version(), sublime.platform(), sublime.architecture()和sublime.channel()之外。

如果插件定义了模块级的方法plugin_loaded(), 会在API ready之后调用。插件也可以定义plugin_unloaded(), 在插件unloaded之前获得到通知。

线程

所有API方法都是线程安全的，不过需要注意，代码运行在次线程上时，应用的状态将会在代码运行时改变。

sublime模块

方法	返回值	描述
set_timeout(callback, delay)	None	在主线程上延时调用(毫秒)． 回调的顺序会按添加的顺序依次执行．
set_async_timeout(callback, delay)	None	在次线程上延时调用(毫秒)。
status_message(string)	None	设置状态栏消息．
error_message(string)	None	显示一个error对话框．
message_dialog(string)	None	显示一个message对话框．
ok_cancel_dialog(string, <ok_button>)	bool	
load_resource(name)	String	载入指定资源。name为Packages/Default/Main.sublime-menu格式
load_binary_resource(name)	bytes	载入指定资源。name需要为。name为Packages/Default/Main.sublime-menu格式．
find_resources(pattern)	[String]	查找name匹配指定模式的资源．
encode_value(value, <pretty>)	String	把JSON对象转换成字符串。如果pretty设置为True，返回的字符串将会包含， 换行和缩进．
decode_value(string)	value	将JSON字符串解码成对象。如果字符串是非法的，会抛出ValueError
load_settings(base_name)	Settings	载入一个配置，name参数要包括文件名和后缀而不是路径。会根据base name搜索插件包，结果返回setting对象。后续调用load_settings载入同一个base_name将返回同一个对象，而不会重新从磁盘读取文件。
save_settings(base_name)	None	保存配置，写入磁盘。
windows()	[Window]	返回打开窗口的列表
active_window()	Window	返回最近使用的一个窗口。
packages_path()	String	返回packages目录的路径．
installed_packages_path()	String	返回所有用户 *.sublime-package文件的目录。
cache_path()	String	返回Sublime Text保存缓存文件的路径。
get_clipboard(<size_limit>)	String	返回剪贴板的内容。size_limit可以防止不必要的大数据，默认16,777,216个字符
set_clipboard(string)	None	设置剪贴板的内容。
score_selector(scope, selector)	Int	把选择器设置成对应的区域，返回区域值。 0标示没有选区，大于0表示有一个选区。 不同的选择器可以通过scope来比较： scope值越高说明这段选区越适合这个选择器．
run_command(string, <args>)	None	运行ApplicationCommand，string是command名字，args是传给command的参数。

log_commands(flag)	None	控制命令的日志。如果启用，所有command从快捷键，菜单中执行都回记录到控制台。
log_input(flag)	None	控制日志输出。如果启用，所有按键都回被记录到控制台。
log_result_regex(flag)	None	控制正则表达式日志。用于调试构建系统中使用的正则表达式比较有用。
version()	String	返回版本号。
platform()	String	返回运行的平台。"osx", "linux" 或者 "windows"。
arch()	String	返回CPU架构。64/32位，"x32" or "x64"。

返回CPU架构。64/32位，"x32" or "x64"。

sublime.View类

view代表了text buffer(缓冲区)中的视图。注意，多个view可以引用同一段buffer，但是它们有自己唯一的选区和几何形状。
|方法|返回值|描述
|—-
|id()|int|返回当前view的唯一标识ID。
|buffer_id()|int|返回当前view下buffer标识的唯一ID。
|file_name()|String|返回buffer关联的完整文件名，如果没有缓冲区存储在磁盘的话返回None。(buffer指缓冲区，下同)
|name()|String|返回buffer指定的名称。
|set_name(name)|None|设置buffer的名称。
|is_loading()|bool|如果buffer还在从磁盘载入返回ture，表示还未准备好给用户使用。
|is_dirty()|bool|返回是否有未保存到buffer的修改。
|is_read_only()|bool|返回true，如果buffer不允许修改。
|set_read_only(value)|None|设置缓冲区不可修改
|is_scratch()|bool|如果缓冲区是临时缓冲区返回True。临时缓冲区不会报告为dirty。
|set_scratch(value)|None|设置buffer为临时缓冲区。
|settings()|Settings|返回view的settings对象。settings对象对当前view是私有的。
|window()|Window|返回持有当前view的window。
|run_command(string, <args>)|None|运行指定的TextCommand，args传入参数。
|size()|int|返回文件中字符总数量。
|substr(region)|String|返回region选区内容字符串。
|substr(point)|String|返回point点的右侧字符。
|insert(edit, point, string)|int|在缓冲区指定的点插入一个字符串。返回插入的字符数量；如果插入当前缓冲区的tabs返回有点区别。
|erase(edit, region)|None|从缓冲区移除region选区内容。
|replace(edit, region, string)|None|把region选区内容替换成指定的字符串。
|sel()|Selection|返回selection(选择)的引用。
|line(point)|Region|返回point点所在的行。
|line(region)|Region|返回region区域行头到行尾的一份拷贝，从行头到行尾可能跨了多行（译者注：换行显示的时候，但是中间没有换行符）。
|full_line(point)|Region|同 line()，但是尾部有换行符的时候也包括了换行符。

|full_line(region)|Region|同 line(), 但是尾部有换行符的时候也包括了换行符
|lines(region)|[Region]|返回region区域的所有行列表（经过排序）。
|split_by_newlines(region)|[Region]|用换行符把整个region分割成多个region区域，返回region列表。
|word(point)|Region|返回包含point点的单词。
|word(region)|Region|返回包含region区域的单词区域（从第一个单词的开头，到最后一个单词的末尾）。有可能会跨多个单词。
|classify(point)|int| 分类pt, 返回0个或多个下面这些标识符按位或的值:

- CLASS_WORD_START
- CLASS_WORD_END
- CLASS_PUNCTUATION_START
- CLASS_PUNCTUATION_END
- CLASS_SUB_WORD_START
- CLASS_SUB_WORD_END
- CLASS_LINE_START
- CLASS_LINE_END
- CLASS_EMPTY_LINE

  |find_by_class(point, forward, classes, <separators>)|Region|从point开始查找下一个匹配classes的位置。如果forward为False，则向后查找。classes是sublime.CLASS_XXX标识符按位或的值。separators可以传入，用于指定哪些字符应该被当成独立的单词。
  |expand_by_class(point, classes, <separators>)|Region|扩展point到左边和右边，直到两端匹配到classes的位置。classes是sublime.CLASS_XXX标识符按位或的值。separators可以传入，用于指定哪些字符应该被当成独立的单词。
  |expand_by_class(region, classes, <separators>)|Region|扩展region到左边和右边，其它参数意义同上。
  |find(pattern, fromPosition, <flags>)|Region|返回匹配的第一个区域，从指定的点位置开始，没有匹配结果返回None。flags参数可以是 sublime.LITERAL, sublime.IGNORECASE, 或者2个"或运算"。
  |find_all(pattern, <flags>, <format>, <extractions>)|[Region]|返回所有(无重叠)的匹配区域结果。flags参数同上， 如果有format参数，所有匹配结果都会按指定格式被格式化并添加到extractions列表里。
  |rowcol(point)|(int, int)|计算指定点从0开始的行位置和列位置。
  |text_point(row, col)|int|计算指定行，列位置字符的偏移量。"col"("列")是从一行的行头开始的字符数量。
  |set_syntax_file(syntax_file)|None|设定view的语法文件。syntax_file是类似Packages/Python/Python.tmLanguage的名称。查看当前view的语法设置view.settings().get('syntax').
  |extract_scope(point)|Region|返回指定点位置字符语法名称的范围。
  |scope_name(point)|String|返回指定点位置字符的语法名称。
  |score_selector(point, selector)|Int|返回包含指定点位置的选择器(selector)的数量(score)。score为0表示没有匹配，大于0表示一个匹配，不同的选择器可以通过scope来比较： scope值越高说明这段选区越适合这个选择器。
  |find_by_selector(selector)|[Regions]|返回符合指定选择器的所有区域，结果为一个列表。
  |show(point, <show_surrounds>)|None|滚动view到指定的点。
  |show(region, <show_surrounds>)|None|滚动view到指定的区域。

|show(region_set, <show_surrounds>)|None|滚动view到可以显示指定的区域集。
|show_at_center(point)|None|滚动到view的中心位置。
|show_at_center(region)|None|滚动view到region区域的中心位置。
|visible_region()|Region|返回当前view可看见的区域。
|viewport_position()|Vector|返回可视区域在布局坐标中的偏移量。
|set_viewport_position(vector, <animate<)|None|把可视区域滚动到指定位置。
|viewport_extent()|vector|返回可视区域宽高。
|layout_extent()|vector|返回文档layout的宽高。(译者注: layout区域相当于编辑器里写的代码的范围，到代码字符的最后一行和最后一列区域，下同)
|text_to_layout(point)|vector|把文本位置转换成layout位置。
|layout_to_text(vector)|point|layout位置转换成文本位置。
|line_height()|real|返回layout的行高。
|em_width()|real|范围layout的字符宽度。
|add_regions(key, [regions], <scope>, <icon>, <flags>)|None| 往view里添加这一组区域(region)。如果region已经存在，会被覆盖。 scope参数决定region绘制的颜色，必须是scope名称，比如 "comment" 或者 "string"。如果没有scope参数，region不会被写入。
icon参数,如果有的话，每个region前面会绘制icon标记。图标的颜色跟scope参数有关。 icon名称可以是: dot、circle,、bookmark,、cross。

可选参数flags可以是下列的组合:

- sublime.DRAW_EMPTY. 用竖线绘制空白区域。默认根本不绘制。
- sublime.HIDE_ON_MINIMAP. 在minimap不显示这些区域。
- sublime.DRAW_EMPTY_AS_OVERWRITE. 用横线绘制空白区域。
- sublime.DRAW_OUTLINED. 绘制区域轮廓而不是填充。
- sublime.PERSISTENT. 保存区域到会话。
- sublime.HIDDEN. 不绘制区域。

下划线样式是唯一的，要么0个要么一个。如果使用underline，需要传入DRAW_NO_FILL和DRAW_NO_OUTLINE。

|get_regions(key)|[regions]|返回指定key的region。
|erase_regions(key)|None|移除指定key的region
|set_status(key, value)|None|往view里添加状态。value值会被现实在状态栏， 以key排序，每个状态值逗号分隔。value为空字符串将清空改key对应的状态值。
|get_status(key)|String|返回key对应的状态值。
|erase_status(key)|None|清空key对应的状态值。
|command_history(index, <modifying_only>)|(String,Dict,int)| 返回undo/redo栈中保存的，命令名称，参数和重复次数。Index 为0 对应最近的一次command， -1对应倒数第二次的命令，一次类推。index为正数代表redo 栈中德命令。如果undo / redo历史记录不足够多返回(None, None, 0) 。如果modifying_only为True (默认为False) 将只会返回修改了缓冲区的输入。
|change_count()|int|返回当前修改的总次数。每次修改时，exclusive都会计数。修改计数可用于判断从上一次期望计数开始是否已经改变。
|fold([regions])|bool|折叠指定区域，如果已经折叠返回False。
|fold(region)|bool|同上。

|unfold(region)|[regions]|展开对应区域的所有文本，返回展开的区域。
|unfold([regions])|[regions]|同上。
|encoding()|String|返回当前文件编码。
|set_encoding(encoding)|None|设置文件编码，文件下一次保存时生效。
|line_endings()|String|返回当前文件使用的换行符模式。
|set_line_endings(line_endings)|None|设置文件的换行符模式，下一次保存时生效。
|overwrite_status()|Bool|返回覆写状态，通常用户通过insert键来切换。
|set_overwrite_status(enabled)|None|设置覆写状态
|symbols(line_endings)|[(Region, String)]|提取所有缓冲区中定义的符号
|show_popup_menu(items, on_done, <flags>)|None| 在光标位置显示一个弹出菜单，用于选择一个列表中的元素。 on_done会调用一次，传入选择项的索引。如果弹出菜单被取消，on_done调用的时候会传入-1参数。Items是一个字符串数组

Flags目前没有选项

sublime.Selection类

维护一组区域(Regions)，确保它们没有重叠。regions按顺序持有。
|方法|返回值|描述
|—-
|clear()|None|Removes all regions.
|add(region)|None|添加指定的region。会与已经存在的有交叉的regions合并。
|add_all(region_set)|None|添加一组regions
|subtract(region)|None|从所有的region中移除指定的region
|contains(region)|bool|如果指定的region是所有region的一个子集则返回true

sublime.Region类

代表了buffer中的一块区域。空白区域可以相等(==)。
|构造器|描述
|—-
|Region(a, b)|指定a,b创建一块区域。

属性	类型	描述
a	int	region区域的第一个结束位置。（译者注：结束位置是相对于整个文档的第一个开始字符而言。）
b	int	region区域的第二个结束位置。b可能会比a小，这样的话就相当于一个反转的区域。
xpos	int	改区域的目标水平位置(target horizontal position)，如果未定义为-1。该值会影响按up和down键时的行为。

方法	返回值	描述
begin()	int	返回a,b中较小的值。
end()	int	返回a,b中较大的值。
size()	int	返回区域的字符总数。始终 >= 0。

empty()	bool	如果begin()==end()，返回True。
cover(region)	Region	返回一个跨越当前region和指定region的一个新的区域。
intersection(region)	Region	返回当前region和指定region的交集
intersects(region)	bool	如果this==region或者当前region和指定region都包含了一个或多个同样的位置。（译者注：其实就是判断指定的region和当前的region是否有交集）
contains(region)	bool	如果指定的region是当前region的一个子集返回True。
contains(point)	bool	如果begin() <= point <= end()返回True。（译者注：point点在当前区域范围内）。

Class sublime.Edit

Edit对象没有方法，它是用于对buffer的修改进行分组。

Edit对象会传给TextCommands，用户不能创建该对象。使用一个非法的Edit对象，或者来自其它view的Edit对象，将会导致引入的方法调用失败。

方法	返回值	描述
(no methods)		

sublime.Window类

方法	返回值	描述
id()	int	返回window的ID.
new_file()	View	创建一个文件。返回一个空的view，view的is_loaded方法返回True。
open_file(file_name, <flags>)	View	打开指定文件，并返回对应的view。如果文件已经被打开，会切换到当前当前视图。注意，文件载入是异步的，view的is_loading() 方法返回False前不能对文件进行操作。

可选参数flags可以是下列的组合:

- sublime.ENCODED_POSITION. 在文件名后面加:row or :row:col 后缀指定打开后的位置
- sublime.TRANSIENT. 只作预览打开文件: 在修改前不会有文件tab分配

  |find_open_file(file_name)|View|在打开的文件列表中查找指定文件，并且返回对应的view,如果没有打开改文件则发挥None
  |active_view()|View|返回当前正在编辑的view。
  |active_view_in_group(group)|View|返回指定组里正在编辑的view。
  |views()|[View]|返回window中所有打开的view。
  |views_in_group(group)|[View]|返回指定组里的所有view。
  |num_groups()|int|返回window中打开的view分组的总数。
  |active_group()|int|返回当前选中组的索引。
  |focus_group(group)|None|激活指定分组。
  |focus_view(view)|None|切换到指定view。

|get_view_index(view)|(group, index)|返回view的分组，和在分组里的索引。如果没有返回-1。
|set_view_index(view, group, index)|None|把view移动到指定分组和指定的索引位置。
|folders()|[String]|返回当前打开的文件夹列表。（译者注：sublime左侧显示的folders列表的每个跟目录）。
|project_file_name()|String|如果有，则返回当前打开的project文件名
|project_data()|Dictionary|返回当前window对应的project数据。内容跟.sublime-project文件的内容一致。
|set_project_data(data)|None| 更新当前window对应的project数据。如果window有对应的.sublime-project文件，将会更新project文件， 同事window也会在内部保存这些数据。
|run_command(string, <args>)|None|运行WindowCommand，传入指定参数。Window.run_command可以运行任何形式的命令，通过输入框来调度命令
|show_quick_panel(items, on_done, <flags>, <selected_index>, <on_highlighted>)|None| 显示一个快速面板，用于选择一个列表中的元素。 on_done会调用一次，传入选择项的索引。如果弹出菜单被取消，on_done调用的时候会传入-1参数。.
Items是一个字符串数组,或者一个字符串数组的二维数组，后者则会在快速面板的每一项显示成多行。

Flags 目前只有一个选项sublime.MONOSPACE_FONT

on_highlighted, 如果指定，每次快速面板上高亮的项变化时都会调用。

|show_input_panel(caption, initial_text, on_done, on_change, on_cancel)|View|显示输入面板，获取一行用户输入。on_done和on_change, 如果不为空的话，需要是接收一个参数的方法。on_cancel是一个不接受任何参数的方法。返回调用输入widget的view
|create_output_panel(name)|View|返回输出面板相对应的view,如果需要则创建它。可以运行show_panel这个window命令来显示输出面板，panel参数设置为一个"output."前缀的name。
|lookup_symbol_in_index(symbol)|[Location]|返回所有定义的符号的位置，当前project下跨文件查找。
|lookup_symbol_in_open_files(symbol)|[Location]|同上功能相似，跨当前打开的文件查找

sublime.Settings类

方法	返回值	描述
get(name)	value	返回指定名称的设置。
get(name, default)	value	返回指定名称的设置，如果没有定义该设置返回默认的。
set(name, value)	None	设置某个名称的配置，只能接受原类型，列表， lists，字典。
erase(name)	None	移除某个配置。如果是继承自父配置不会被删除。
has(name)	bool	判断当前配置类中是否存在某个配置或者父配置中是否存在。
add_on_change(key, on_change)	None	注册当前配置对象的change的回调。只要有一个配置发生变化都会被回调。.
clear_on_change(key)	None	移除指定的change回调。

Module sublime_plugin

方法	返回值	描述
(no methods)		

Class sublime_plugin.EventListener

注意，有许多事件是view下的buffer缓冲区触发的，而且这些方法只调用一次， view作为第一个参数。
|方法|返回值|描述
|—-
|on_new(view)|None|当创建一个新的buffer时触发。
|on_new_async(view)|None|同上，运行在一个独立的线程上，这样就不中断主应用。
|on_clone(view)|None|当从一个已存在的view复制一份时触发。
|on_clone_async(view)|None|同on_new_async原理
|on_load(view)|None|当文件载入完成时触发。
|on_load_async(view)|None|同on_new_async原理
|on_pre_close(view)|None|在一个view关闭前触发。
|on_close(view)|None|在一个view关闭时触发 (注意，同一个缓冲区还可能有其它view)。
|on_pre_save(view)|None|在一个view保存前触发。
|on_pre_save_async(view)|None|同on_new_async原理
|on_post_save(view)|None|在一个view保存后触发。
|on_post_save_async(view)|None|同on_new_async原理
|on_modified(view)|None|view被修改后触发。
|on_modified_async(view)|None|同on_new_async原理
|on_selection_modified(view)|None|view里的选区变化时触发。
|on_selection_modified_async(view)|None|同on_new_async原理
|on_activated(view)|None|一个view被激活时触发。
|on_activated_async(view)|None|同on_new_async原理
|on_deactivated(view)|None|一个view被隐藏时触发（被切换到后台）。
|on_deactivated_async(view)|None|同on_new_async原理
|on_text_command(view, command_name, args)|(new_command_name, new_args)|当一个text command触发时调用。监听者需要返回一个元祖(command, arguments) 用于重写这个command，或者不做任何修改时返回None。
|on_window_command(window, command_name, args)|(new_command_name, new_args)|当一个window command触发时调用。用法同上
|post_text_command(view, command_name, args)|None|当一个text command执行之后调用。
|post_window_command(window, command_name, args)|None|当一个window command执行之后调用。
|on_query_context(view, key, operator, operand, match_all)|bool or None| 当指定的key触发键盘绑定时调用。如果该插件知道如何响应这个上下文，应该返回一个虚假的True，如果不识别该上下文则返回None。

operator可以是下面这些值:

- sublime.OP_EQUAL. 上下文是否与operand相等?
- sublime.OP_NOT_EQUAL. 上下文是否与operand不相等?
- sublime.OP_REGEX_MATCH. 上下文与operand指定的正则匹配?
- sublime.OP_NOT_REGEX_MATCH. 上下文与operand指定的正则不匹配?
- sublime.OP_REGEX_CONTAINS. 上下文包含与operand指定的正则相匹配的子字符串?

- sublime.OP_NOT_REGEX_CONTAINS.上下文不包含与operand指定的正则相匹配的子字符串
  match_all 当与选区相关时使用: 是否所有的选区都要匹配(match_all = True), 还是只需要至少一个选区匹配(match_all = Fals)?

sublime_plugin.ApplicationCommand类

方法	返回值	描述
run(<args>)	None	当command运行时执行。
is_enabled(<args>)	bool	如果command在当前时间可运行返回True。 默认实现都返回Flase。
is_visible(<args>)	bool	如果command在当前可显示在菜单。默认实现都返回False。
is_checked(<args>)	bool	如果需要在菜单项旁边显示checkbox则返回true。.sublime-menu文件必须要有checkbox属性，设置为true
description(<args>)	String	返回command的描述。在菜单中使用，如果没有标题的情况下。返回None获取默认描述。

sublime_plugin.WindowCommand类

WindowCommands 每个window只初始化一次。Window对象可以通过self.window来引用。
|方法|返回值|描述
|—-
|run(<args>)|None|command运行时调用。
|is_enabled(<args>)|bool|如果command在当前时间可运行返回True。 默认实现都返回Flase。
|is_visible(<args>)|bool|如果command在当前可显示在菜单。默认实现都返回False。
|description(<args>)|String|返回command的描述。在菜单中使用，如果没有标题的情况下。返回None获取默认描述。

sublime_plugin.TextCommand类

TextCommands每个view只初始化一次。可以通过self.view放访问当前view。
|方法|返回值|描述
|—-
|run(edit, <args>)|None|command运行时调用。
|is_enabled(<args>)|bool|如果command在当前时间可运行返回True。 默认实现都返回Flase。
|is_visible(<args>)|bool|如果command在当前可显示在菜单。默认实现都返回False。
|description(<args>)|String|返回command的描述。在菜单中使用，如果没有标题的情况下。返回None获取默认描述。

原文:

> *http://feliving.github.io/Sublime-Text-3-Documentation/api_reference.html*

迁移指南

迁移指南

概述

Sublime Text 3在插件上与Sublime Text 2相比有很大的不同，大多数的插件至少都需要做少量的迁移工作。有下面这些变化：

- Python 3.3
- Out of Process Plugins(进程插件)
- Asynchronous Events(匿名事件)
- 受限制的begin_edit()和end_edit()
- Zip压缩的Packages
- Importing Modules(引入模块)

Python 3.3

Sublime Text 3使用Python 3.3，而Sublime Text 2使用的是Python 2.6。此外，OS X系统下，不在使用系统的Python构建工具，而使用Sublime Text自己的。Windows和Linux跟原来一样绑定到自己的版本。

Out of Process Plugins

插件现在运行在一个独立的进程，plugin_host。从插件作者的角度来看没什么区别，除了plugin host崩溃不再会使主程序挂掉之外。

匿名事件

在Sublime Text 2中仅有set_timeout方法是线程安全的。在Sublime Text 3中，每个API方法都是线程安全的。另外，还有匿名事件处理器，可以更容易编写不中断的代码：

- on_modified_async
- on_selection_modified_async
- on_pre_save_async
- on_post_save_async
- on_activated_async
- on_deactivated_async
- on_new_async
- on_load_async
- on_clone_async
- set_timeout_async
  当编写线程代码时，需要注意在方法运行时缓冲区可能会随时改变。

受限制的begin_edit()和end_edit()

begin_end()和end_edit()不再是可以直接访问，除了在某些特定情况下。获取一个有效的编辑器对象的唯一方法是把你的代码放在TextCommand的一个子类里。通常，TextCommand中大部分代码可以放在begin_edit()和end_edit()之间进行重构，然后在这个TextCommand上调用run_command。

这种方式可以消除空荡的Edit对象，并且确保重复命令和宏正常工作。

Zip压缩的包

Sublime Text 3中的Packages可以直接通过.sublime-package格式文件运行(其实就是.zip重命名的格式)，而Sublime Text 2是未zip压缩运行的。

跟大多数改变相比这不会导致什么变化，重要的一点是，你要访问packgae中的文件时需要注意。

Importing Modules

引入其他插件在Sublime Text 3中会更加的简单和稳定,而且可以使用规整的引入声明，例如import Default.comment将会引入Packages/Default/Comment.py

启动时API使用会受限制

由于plugin_host载入是异步的，在载入时Sublime Text的 API将不可用。这就意味着你的模块里所有top-level的声明都不能调用sublime模块的方法。在启动期间，API是休眠状态，将忽略所有调用请求。

原文:

> *http://feliving.github.io/Sublime-Text-3-Documentation/porting_guide.html*