

TOPICS Basic Editing ▼

基本编辑



(https://github.com/Microsoft/vscode-docs/blob/master/docs/editor/codebasics.md)

Visual Studio Code首先是一个编辑器，它包含高效编辑源代码所需的功能。本主题带您了解编辑器的基础知识，并可以帮助您逐步使用代码。

键盘快捷键

编写代码时能够将手放在键盘上对于提高生产率至关重要。VS Code具有丰富的默认键盘快捷键集，并且允许您自定义它们。

- 键盘快捷方式参考 (/docs/getstarted/keybindings#_keyboard-shortcuts-reference) -通过下载参考表了解最常用和流行的键盘快捷方式。
- 安装Keymap扩展名 (/docs/getstarted/keybindings#_keymap-extensions) -通过安装Keymap扩展名，在VS Code中使用旧编辑器的键盘快捷键（例如Sublime Text，Atom和Vim）。
- 自定义键盘快捷键 (/docs/getstarted/keybindings#_customizing-shortcuts) -更改默认键盘快捷键以适合您的样式。

多种选择（多光标）

VS Code支持多个光标，可以快速同时进行编辑。您可以使用 Alt + Click 添加辅助光标（渲染更细）。每个游标根据其所在的上下文独立运行。添加更多游标的常用方法是使用 Ctrl + Alt + Down 或 Ctrl + Alt + Up 在下方或上方插入光标。

注意：您的图形卡驱动程序（例如NVIDIA）可能会覆盖这些默认快捷方式。

```
31 .global-message-list.transition {
32 → -webkit-transition: top 200ms linear;
33 → -ms-transition:      top 200ms linear;
34 → -moz-transition:     top 200ms linear;
35 → -khtml-transition:   top 200ms linear;
36 → -o-transition:       top 200ms linear;
37 → transition:          top 200ms linear;
38 }
```

Ctrl + D 选择光标处的单词，或当前选择的下一个出现的单词。

```
6
7 The quick brown fox jumps over the lazy dog.
8 → The quick brown fox jumps over the lazy dog.
9 → → The quick brown fox jumps over the lazy dog.
10 → → → The quick brown fox jumps over the lazy dog.
11 → → → → The quick brown fox jumps over the lazy dog.
12 → → → → → The quick brown fox jumps over the lazy dog.
13
14
```

提示：您还可以使用 Ctrl + Shift + L 添加更多光标，这将在每次出现当前所选文本时添加一个选择。

多光标修饰符

如果您想更改将多个光标应用于macOS上的 Cmd + Click 以及Windows和Linux上的 Ctrl + Click 的修饰键，则可以使用 editor.multiCursorModifier 设置来完成 (/docs/getstarted/settings)。这样，来自其他编辑器（例如Sublime Text或Atom）的用户就可以继续使用他们熟悉的键盘修改器。

该设置可以设置为：

- ctrlCmd - 在Windows 上映射为 Ctrl，在macOS 上映射为 Cmd。
- alt - 现有的默认 Alt。

在“选择”菜单中还有一个菜单项“使用Ctrl +单击多光标”可快速切换此设置。

“转到定义”和“打开链接”手势也将遵守此设置并进行调整，以使它们不会冲突。例如，当设置 ctrlCmd 为时，可以使用 Ctrl / Cmd + Click 添加多个光标，并且可以使用 Alt + Click 调用打开链接或转到定义。

缩小/展开选择

快速缩小或扩展当前选择。使用 Shift + Alt + Left 和 Shift + Alt + Right 触发它。

这是使用 Shift + Alt + Right 扩展选择范围的示例：

```
7 namespace Hello1
8 {
    0 references
9     class Program
10    {
        0 references
11        static void Main(string[] args)
12        {
13            System.Console.WriteLine("Hello World!");
14        }
15    }
16 }
```

列（框）选择

将光标放在一个角上，然后在按住 Shift + Alt的 同时拖动到对角：

```
208
209 // Key                Character                Virtual Key Code
210 // Semicolon           ';'                    186
211 // Colon               ':'                    186
212 // Equals sign         '='                    187
213 // Plus                '+'                    187
214 // Comma               ','                    188
215 // Less than sign      '<'                    188
216 // Minus               '-'                    189
217 // Underscore          '_'                    189
218 // Period              '.'                    190
219 // Greater than sign   '>'                    190
220 // Forward slash       '/'                    191
221
```

注意：当使用 Ctrl / Cmd 作为多光标修改器时，这将更改为 Shift + Ctrl / Cmd 。

在macOS和Windows上，还有用于列选择的默认键绑定，但在Linux上没有。

键	命令	命令ID
Ctrl + Shift + Alt +向下	列选择向下	cursorColumnSelectDown
Ctrl + Shift + Alt +向上键	列选择	cursorColumnSelectUp
Ctrl + Shift + Alt +向左	列选择左	cursorColumnSelectLeft
Ctrl + Shift + Alt +向右键	列选择权	cursorColumnSelectRight
Ctrl + Shift + Alt + PageDown	列选择向下	cursorColumnSelectPageDown
Ctrl + Shift + Alt + PageUp	列选择上一页	cursorColumnSelectPageUp

您可以编辑 (/docs/getstarted/keybindings)您 keybindings.json 将它们绑定的东西，如果你想更熟悉。

保存/自动保存

默认情况下，VS代码需要一个明确的行动，将更改保存到磁盘， 按Ctrl + S 。

但是，打开它很容易 Auto Save ， 它会在配置的延迟或焦点离开编辑器后保存您的更改。启用此选项后，无需显式保存文件。打开文件的最简单方法 Auto Save 是使用“文件” > “自动保存”切换，该按钮在延迟后打开和关闭保存。

要进一步控制 Auto Save ， 请打开“用户”或“工作区” 设置， (/docs/getstarted/settings)然后找到相关的设置：

- files.autoSave： 可以具有以下值：
 - off -禁用自动保存。
 - afterDelay -在配置的延迟（默认1000毫秒）后保存文件。
 - onFocusChange -当焦点移出脏文件的编辑器时保存文件。
 - onWindowChange -当焦点移出“ VS代码”窗口时保存文件。
- files.autoSaveDelay： 当 files.autoSave 配置为时，配置延迟（以毫秒为单位） afterDelay 。默认值为1000毫秒。

热出口

默认情况下，VS Code会记住未保存的文件更改。当通过文件 > 退出（在macOS上为代码 > 退出）关闭应用程序或关闭最后一个窗口时，触发热退出。

您可以通过设置 files.hotExit 以下值来配置热出口：

- "off"： 禁用热出口。
- "onExit"： 当应用程序关闭时，即Windows / Linux上的最后一个窗口关闭时， 或当 workbench.action.quit 命令被触发时（从**命令面板**， 键盘快捷键或菜单）， 将触发热退出。所有未打开文件夹的窗口将在下次启动时恢复。
- "onExitAndWindowClose"： 当应用程序关闭时，即在Windows / Linux上关闭最后一个窗口时， 或者当 workbench.action.quit 命令被触发时（从**命令面板**， 键盘快捷键或菜单）， 以及具有文件夹的任何窗口被关闭时， 都会触发热退出。无论是否是最后一个窗口都将打开。所有未打开文件夹的窗口将在下次启动时恢复。要恢复关闭之前的文件夹窗口， 请设置 window.restoreWindows 为 all 。

如果热退出发生问题，所有备份都存储在以下文件夹中，用于标准安装位置：

- **视窗** %APPDATA%\Code\Backups
- **苹果系统** \$HOME/Library/Application Support/Code/Backups
- **的Linux** \$HOME/.config/Code/Backups

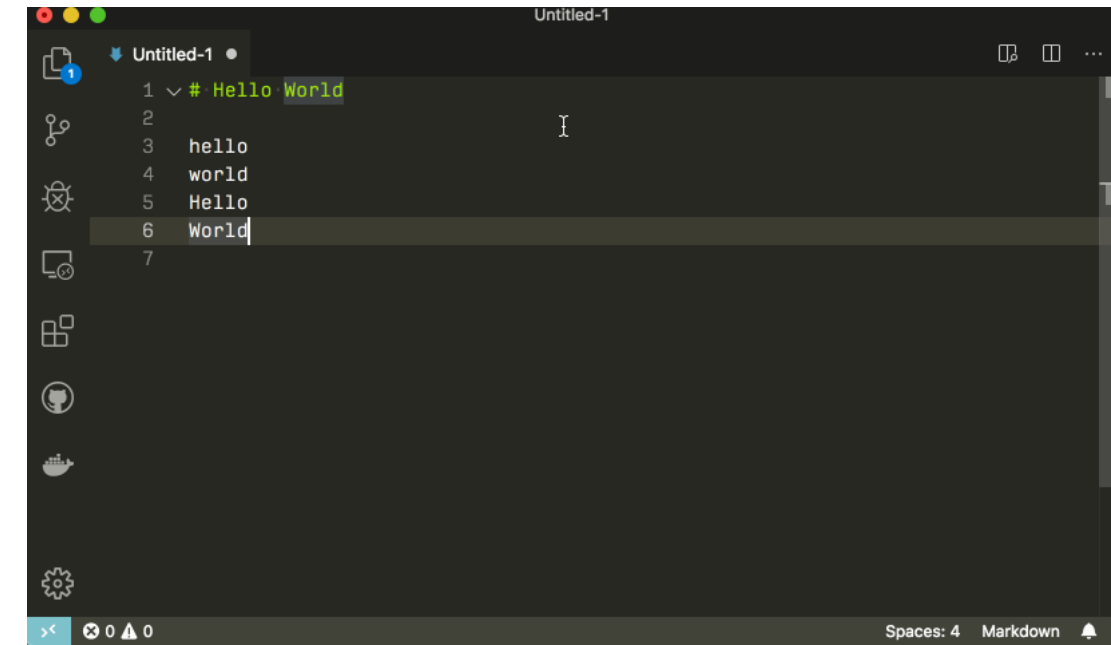
查找和替换

VS Code允许您快速查找文本并替换为当前打开的文件。按 `Ctrl + F` 在编辑器中打开“查找小部件”，搜索结果将在编辑器，概述标尺和小地图中突出显示。

如果当前打开的文件中有多个匹配结果，则可以在查找输入框处于焦点状态时按 `Enter` 和 `Shift + Enter` 导航到下一个或上一个结果。

选择中的种子搜索字符串

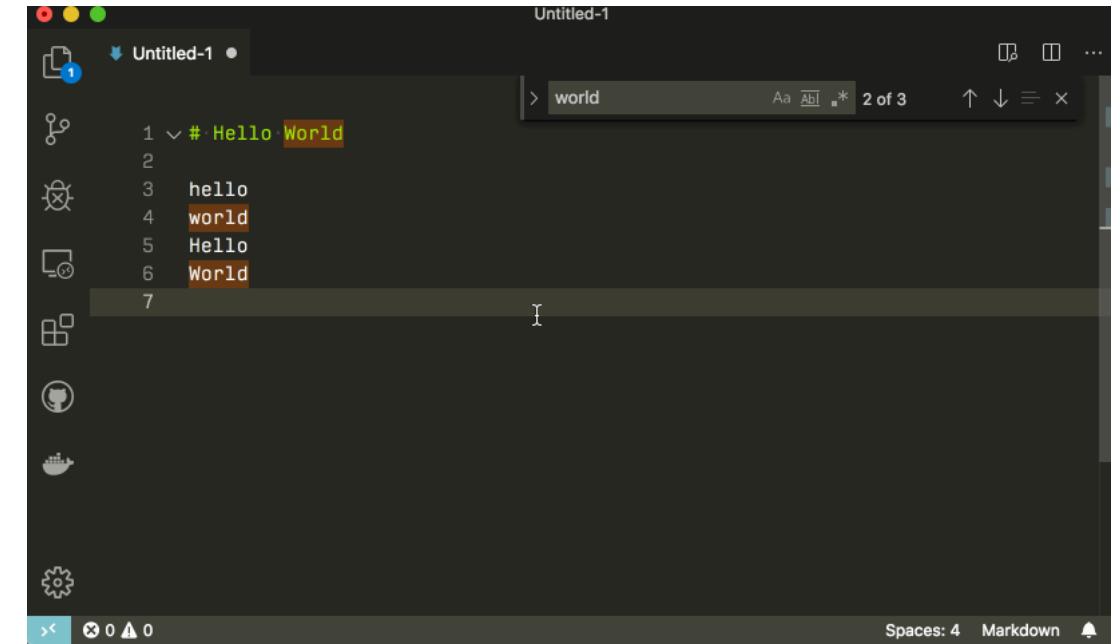
打开“查找小部件”后，它将自动在编辑器中将所选文本填充到“查找”输入框中。如果选择为空，则会将光标下方的单词插入到输入框中。



可以通过设置 `editor.find.seedSearchStringFromSelection` 为 `false` 来关闭此功能。

查找选择

默认情况下，查找操作在编辑器中的整个文件上运行。它也可以在选定的文本上运行。您可以通过单击“查找”小部件上的汉堡包图标来启用此功能。



如果希望它是“查找小部件”的默认行为，则仅当选择了多行内容时可以在选定的文本上运行，则可以设置 `editor.find.autoFindInSelection` 为 `always` 或 `multiline`。

高级查找和替换选项

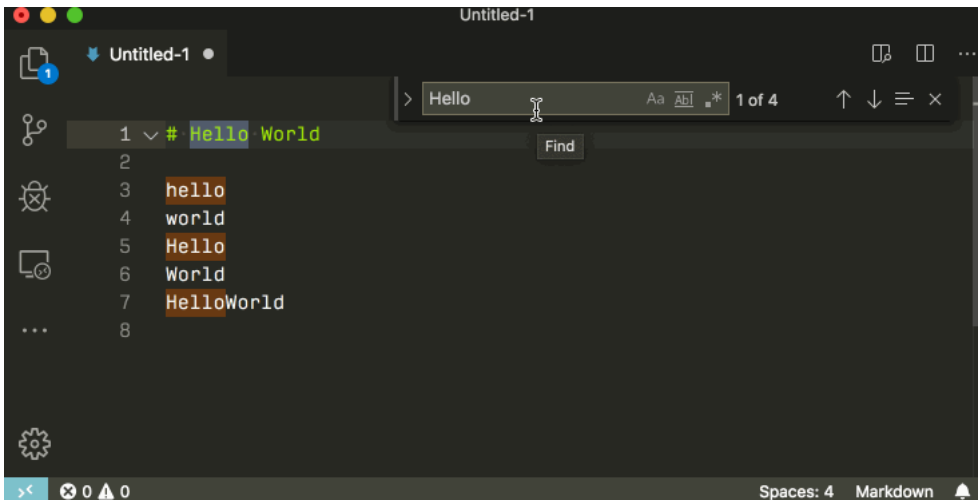
除了查找和替换为纯文本之外，“查找小部件”还具有三个高级搜索选项：

- 相符
- 匹配整个单词
- 正则表达式

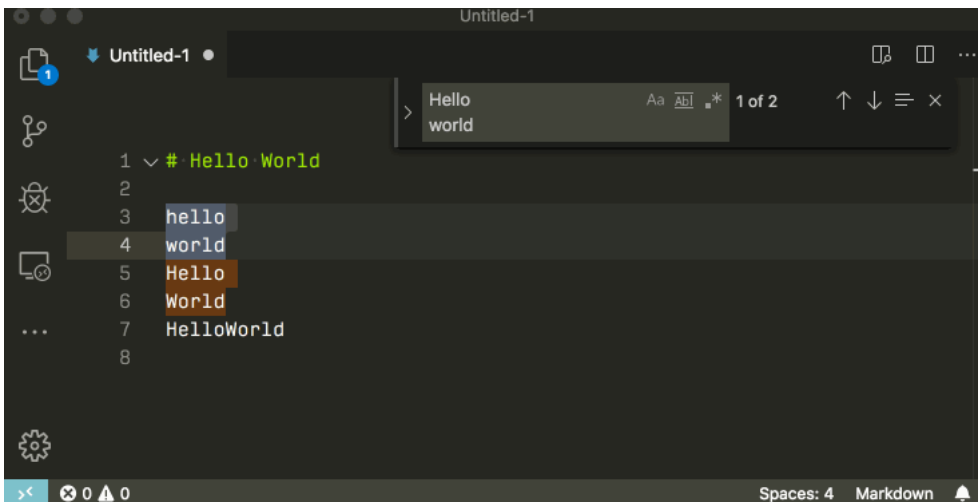
替换输入框支持大小写保留，您可以通过单击保留大小写（`AB`）按钮将其打开。

多行支持和Find Widget调整大小

您可以通过将文本粘贴到“查找”输入框和“替换”输入框中来搜索多行文本。按下 `Ctrl+Enter` 将在输入框中插入新行。

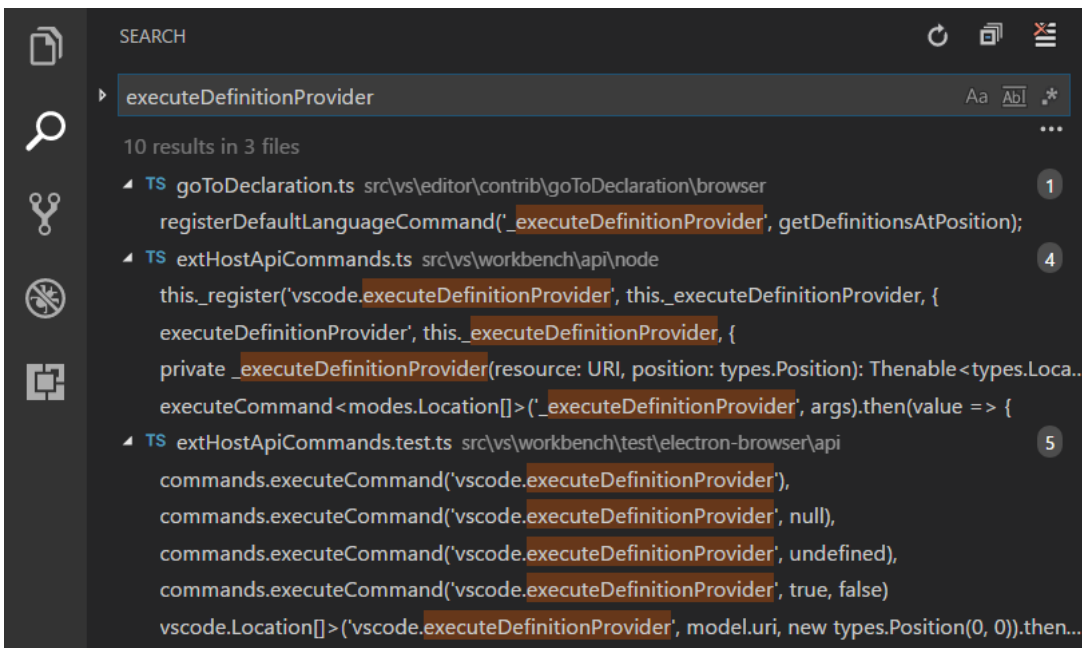


搜索长文本时，“查找小部件”的默认大小可能太小。您可以拖动左窗框以放大“查找小部件”，或双击左窗框以将其最大化或缩小到其默认大小。



搜索文件

VS Code允许您快速搜索当前打开的文件夹中的所有文件。按 `Ctrl + Shift + F` 并输入搜索词。搜索结果被分组到包含搜索词的文件中，并标明每个文件中的匹配项及其位置。展开文件以查看该文件中所有匹配的预览。然后单击其中一个命中以在编辑器中查看它。



提示： 我们也支持在搜索框中进行正则表达式搜索。

您可以通过单击右侧搜索框下方的省略号（或**切换搜索详细信息**）来配置高级搜索选项（或按 `Ctrl + Shift + J`）。这将显示其他字段以配置搜索。

进阶搜寻选项



在搜索框下方的输入框中，您可以输入要包含在搜索中或从搜索中排除的模式。如果输入 `example`，则将匹配 `example` 工作空间中命名的每个文件夹和文件。如果输入 `./example`，则将匹配 `example/` 工作空间顶层的文件夹。用于！从搜索中排除那些模式。`!example` 将跳过搜索名为的任何文件夹或文件 `example`。使用 `,` 分隔多个模式。路径必须使用正斜杠。您还可以使用glob语法：

- `*` 匹配路径段中的一个或多个字符
- `?` 匹配路径段中的一个字符
- `**` 匹配任意数量的路径段，包括无
- `{}` 对条件进行分组（例如，`{**/*.html,**/*.txt}` 匹配所有HTML和文本文件）
- `[]` 申报范围的字符来匹配（`example.[0-9]` 匹配上 `example.0`，`example.1 ...`）

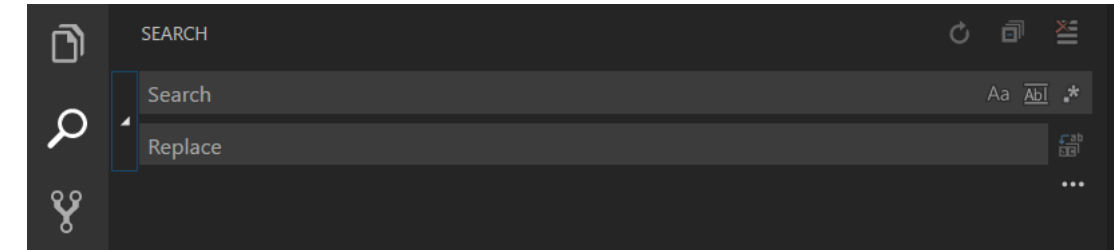
VS Code默认情况下会排除某些文件夹，以减少您不感兴趣的搜索结果的数量（例如：）`node_modules`。打开设置 (`/docs/getstarted/settings`)以在 `files.exclude` 和 `search.exclude` 部分下更改这些规则。

还要注意**要排除的文件**框中的“**使用排除设置和忽略文件**”切换按钮。切换按钮确定是否排除文件忽略的文件和/或您的和设置匹配的文件。`.gitignore` `files.exclude` `search.exclude`

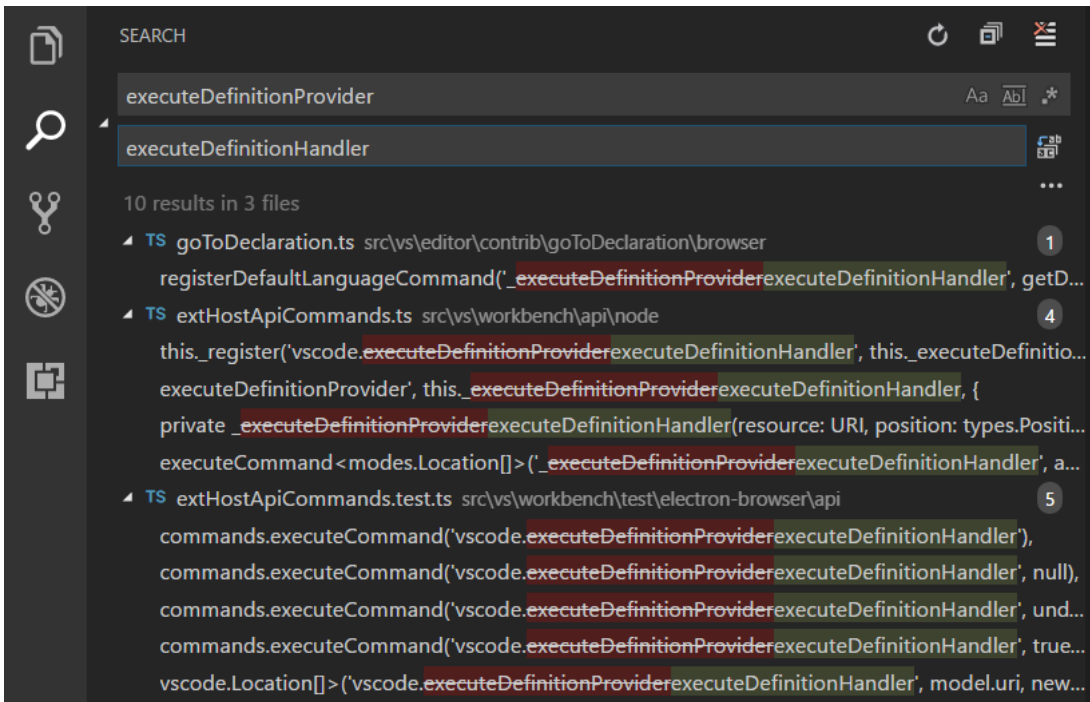
提示：在资源管理器中，您可以右键单击文件夹，然后选择“**在文件夹中查找**”以**仅在文件夹内**搜索。

搜索并替换

您也可以跨文件搜索和替换。展开搜索窗口小部件以显示替换文本框。



在“替换”文本框中键入文本时，您将看到差异显示未完成的更改。您可以从“替换”文本框中替换所有文件，将所有文件替换为一个文件或替换单个更改。



提示：通过使用 向下 和 向上 浏览搜索词历史记录，您可以快速重用上一个搜索词。

智能感知

我们将始终提供单词补全功能，但是对于丰富的语言 (/docs/languages/overview) (例如JavaScript, JSON, HTML, CSS, SCSS, Less, C# 和TypeScript)，我们提供了真正的IntelliSense体验。如果语言服务知道可能的补全，则在您键入时会弹出IntelliSense建议。您始终可以使用 Ctrl + Space 手动触发它。默认情况下，Tab 或 Enter 是接受键盘触发器，但是您也可以自定义这些键绑定 (/docs/getstarted/keybindings)。

提示：建议过滤支持CamelCase，因此您可以在方法名称中输入大写字母以限制建议。例如，“cra”将快速启动“createApplication”。

提示：可以通过 editor.quickSuggestions 和 editor.suggestOnTriggerCharacters 设置 (/docs/getstarted/settings) IntelliSense建议。

JavaScript和TypeScript开发人员可以利用npmjs (<https://www.npmjs.com>)类型声明 (类型) 文件存储库来获取通用JavaScript库 (Node.js, React, Angular) 的IntelliSense。您可以在JavaScript语言 (/docs/languages/javascript#_intellisense)主题和Node.js (/docs/nodejs/nodejs-tutorial)教程中找到有关使用类型声明文件的很好的解释。

在IntelliSense文档中 (/docs/editor/intellisense)了解更多信息 (/docs/editor/intellisense)。

格式化

VS Code对源代码格式提供了极大的支持。编辑器有两个明确的格式操作：

- **格式化文档** (Shift + Alt + F) -格式化整个活动文件。
- **格式选择** (Ctrl + K Ctrl + F) -格式化所选文本。

您可以从**命令面板** (Ctrl + Shift + P) 或编辑器上下文菜单中调用它们。

VS Code具有JavaScript, TypeScript, JSON和HTML的默认格式化程序。每种语言都有特定的格式设置选项 (例如 `html.format.indentInnerHtml`)，您可以在用户或工作空间设置 (/docs/getstarted/settings)中将其调整为自己的偏好。如果安装了另一个扩展程序可以提供相同语言的格式设置，则也可以禁用默认语言格式程序。

```
"html.format.enable": false
```

除了手动调用代码格式外，您还可以根据用户手势 (例如键入，保存或粘贴) 来触发格式。这些默认情况下处于关闭状态，但是您可以通过以下设置 (/docs/getstarted/settings)启用这些行为：

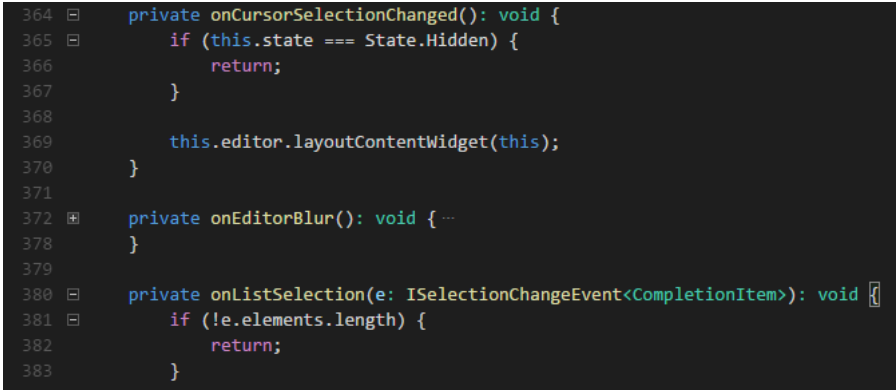
- `editor.formatOnType` -键入后设置行格式。
- `editor.formatOnSave` -保存时格式化文件。
- `editor.formatOnPaste` -格式化粘贴的内容。

注意：并非所有格式化程序都支持粘贴格式，因此它们必须支持格式化所选内容或文本范围。

除了默认的格式化程序外，您还可以在Marketplace上找到扩展名以支持其他语言或格式化工具。有一个 `Formatters` 类别，您可以轻松搜索和查找格式扩展名 (<https://marketplace.visualstudio.com/search?target=VSCode&category=Formatters&sortBy=Downloads>)。在“**扩展名**”视图搜索框中，键入“`formatters`”或“`category: formatters`”以查看VS Code中扩展名的过滤列表。

折叠式

您可以使用装订线在行号和行首之间的折叠图标来折叠源代码区域。将鼠标移到排水沟上，然后单击以折叠和展开区域。使用 Shift +单击 折叠图标可折叠或展开区域及其内部的所有区域。



您还可以使用以下操作：

- 折叠 （ Ctrl + Shift + [） 在光标处折叠最里面的未折叠区域。
- 展开 （ Ctrl + Shift +] ） 会 展开光标处的折叠区域。
- Toggle Fold （ Ctrl + K Ctrl + L ） 折叠或展开光标处的区域。
- 递归折叠 （ Ctrl + K Ctrl + [） 折叠光标最里面的未折叠区域以及该区域内的所有区域。
- 递归展开 （ Ctrl + K Ctrl +] ） 展开光标处的区域以及该区域内的所有区域。
- 全部折叠 （ Ctrl + K Ctrl + 0 ） 折叠编辑器中的所有区域。
- 全部展开 （ Ctrl + K Ctrl + J ） 展开编辑器中的所有区域。
- 折叠级别X（级别2的 Ctrl + K Ctrl + 2 ） 折叠级别X的所有区域，除了当前光标位置处的区域。
- 折叠所有块注释 （ Ctrl + K Ctrl + / ） 折叠以块注释标记开头的所有区域。

默认情况下，折叠区域是基于线条的缩进来评估的。当一条线的缩进小于一条或多条后续线时，折叠区域开始；当一条线的缩进相同或较小时，折叠区域结束。从1.22版本开始，还可以基于编辑器配置语言的语法标记来计算折叠区域。下列语言已提供语法识别折叠：Markdown，HTML，CSS，LESS，SCSS和JSON。

如果您希望针对上述一种（或全部）语言切换回基于缩进的折叠，请使用：

```
"[html]": {
  "editor.foldingStrategy": "indentation"
},
```

区域也可以通过每种语言定义的标记来定义。当前，以下语言已定义了标记：

语言	起始区域	末端区域
蝙蝠	::#region 要么 REM #region	::#endregion 要么 REM #endregion
C #	#region	#endregion
C / C ++	#pragma region	#pragma endregion
CSS / Less / SCSS	/*#region*/	/*#endregion*/
咖啡脚本	#region	#endregion
F #	//#region 要么 (#_region)	//#endregion 要么 (#_endregion)
爪哇	//#region 要么 //<editor-fold>	// #endregion 要么 //</editor-fold>
降价促销	<!-- #region -->	<!-- #endregion -->
Perl5	#region 要么 =pod	#endregion 要么 =cut
的PHP	#region	#endregion
电源外壳	#region	#endregion
蟒蛇	#region 要么 # region	#endregion 要么 # endregion
TypeScript / JavaScript	//#region	//#endregion
Visual Basic	#Region	#End Region

要仅折叠和展开标记定义的区域，请使用：

- 折叠标记区域 （ Ctrl + K Ctrl + 8 ） 折叠所有标记区域。
- 展开标记区域 （ Ctrl + K Ctrl + 9 ） 展开所有标记区域。

缩进

VS Code可让您控制文本的缩进以及是否要使用空格或制表位。默认情况下，VS Code插入空格，每个 Tab 键使用4个空格。如果您想使用其他默认设置，则可以修改 editor.insertSpaces 和 editor.tabSize 设置 (/docs/getstarted/settings)。

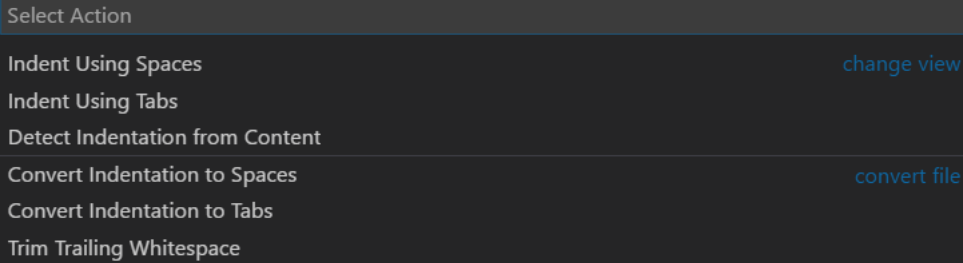
```
"editor.insertSpaces": true,  
"editor.tabSize": 4,
```

自动侦测

VS Code分析打开的文件并确定文档中使用的缩进。自动检测到的缩进将覆盖您的默认缩进设置。检测到的设置显示在状态栏的右侧：

Ln 1, Col 1 **Tab Size: 4** UTF-8 CRLF

您可以单击状态栏缩进显示以显示带有缩进命令的下拉菜单，允许您更改打开文件的默认设置或在制表位和空格之间进行转换。



注意： VS Code自动检测检查2、4、6或8个空格的缩进。如果文件使用不同数量的空格，则可能无法正确检测到缩进。例如，如果您的约定是缩进3个空格，则可能需要关闭 `editor.detectIndentation` 并将制表符大小显示设置为3。

```
"editor.detectIndentation": false,  
"editor.tabSize": 3,
```

文件编码支持

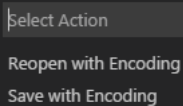
使用“**用户设置**”或“**工作区设置**”中的 `files.encoding` 设置全局或按工作区设置文件编码。

```
//----- Files configuration -----  
  
// The default character set encoding to use when reading and writing files.  
"files.encoding": "utf8",
```

您可以在状态栏中查看文件编码。

Ln 11, Col 1 **UTF-8** CRLF JavaScript

单击状态栏中的编码按钮，以使用其他编码重新打开或保存活动文件。



然后选择一种编码。



下一步

您已经介绍了基本的用户界面-VS Code还有很多内容。继续阅读以了解以下内容：

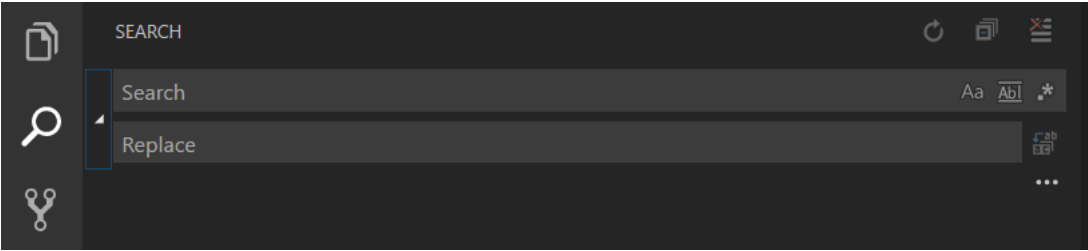
- 简介视频-设置和基础 (/docs/introvideos/basics) - 观看有关VS Code基础的教程。
- 用户/工作区设置 (/docs/getstarted/settings) - 了解如何通过用户和工作区设置根据自己的喜好配置VS Code。
- 代码导航 (/docs/editor/editingevolved) - 窥视和转到定义等。
- 集成终端 (/docs/editor/integrated-terminal) - 了解用于在VS Code中快速执行命令行任务的集成终端。
- IntelliSense (/docs/editor/intellisense) - VS Code带来了智能代码补全功能。

- 调试 (/docs/editor/debugging) -这是VS Code真正发挥作用的地方。

常见问题

是否可以全局搜索和替换？ #

是的，展开搜索视图文本框以包含替换文本字段。您可以搜索和替换工作空间中的所有文件。请注意，如果您没有在文件夹上打开VS Code，则搜索将仅在当前打开的文件上运行。



如何打开自动换行？

您可以通过 `editor.wordWrap` 设置 (/docs/getstarted/settings)控制自动换行。默认情况下 `editor.wordWrap` 为 `is, off` 但如果将其设置为 `on`，则文本将环绕在编辑器的视口宽度上。

```
"editor.wordWrap": "on"
```

您可以使用 `Alt + Z` 切换VS Code会话的自动换行。

您还可以使用该 `editor.rulers` 设置将垂直列标尺添加到编辑器中，该设置会在您想要垂直标尺的位置获取一系列列字符位置。

该文档对您有帮助吗？

是 没有