VS Code

configuration file

Inside `.vscode` file, change `tasks.json` to run tasks, change `launch.json` to run the debugger

Change `settings.json` to overwrite user settings.

Change `keybindings.json` to customize keyboard shortcuts.

`.eslintrc.json`

Install the **ESLint extension**. Configure your linter however you'd like. Consult the ESLint specification for details on its linting rules and options.

`package.json`

See IntelliSense for your `package.json` file.

`setting.json`

some of my settings

```
{
    "editor.fontSize": 14,
    //"terminal.integrated.shell.windows": "C:\\WINDOWS\\System32\\cmd.exe",
    "terminal.integrated.shell.windows": "C:\\extension\\Git\\bin\\bash.exe",
    "terminal.external.windowsExec": "C:\\extension\\Git\\bin\\bash.exe",


    "workbench.colorTheme": "Solarized Light",
    "[markdown]": {
        "editor.wordWrap": "on",
        "editor.quickSuggestions": true
    },
    "git.path": "C:\\extension\\Git\\bin.exe",
    "editor.suggestSelection": "first",
    "vsintellicode.modify.editor.suggestSelection":
"automaticallyOverrodeDefaultValue",
    "python.jediEnabled": false,
    "terminal.integrated.inheritEnv": false, //I don't want to use terminal for Anaconda
    "diffEditor.renderSideBySide": false,// version control for diffs is not side by side but a inline view
    "python.pythonPath": "C:\\software\\Anaconda3\\python.exe",
}
```

check for more customizations here

For the settings, which you only want for specific languages, you can scope the settings by the language identifier. You can find a list of commonly used language ids in the Language Identifiers reference.

```
"[languageid]": {

}
```

Tips and Tricks

Getting started

Open the **Welcome** page to get started with the basics of VS Code. **Help > Welcome**.

Open multiple files from Quick Open

You can open multiple files from **Quick Open** by pressing the Right arrow key. This will open the currently selected file in the background and you can continue selecting files from **Quick Open**. (在同一个文件夹内)

Change language mode

If you want to persist the new language mode for that file type, you can use the **Configure File Association for** command to associate the current file extension with an installed language.

File associations

Create language associations for files that aren't detected correctly. For example, many configuration files with custom file extensions are actually JSON.

```
"files.associations": {
    ".database": "json"
}
```

Preventing dirty writes

VS Code will show you an error message when you try to save a file that cannot be saved because it has changed on disk. VS Code blocks saving the file to prevent overwriting changes that have been made outside of the editor.

In order to resolve the save conflict, click the **Compare** action in the error message to open a diff editor that will show you the contents of the file on disk (to the left) compared to the contents in VS Code (on the right):

Use the actions in the editor toolbar to resolve the save conflict. You can either **Accept** your changes and thereby overwriting any changes on disk, or **Revert** to the version on disk. Reverting means that your changes will be lost.

Note: The file will remain dirty and cannot be saved until you pick one of the two actions to resolve the conflict.

Keymaps

Are you used to keyboard shortcuts from another editor? You can install a Keymap extension that brings the keyboard shortcuts from your favorite editor to VS Code. Go to **Preferences > Keymap Extensions** to see the current list on the Marketplace.

Portable mode

VS Code has a Portable mode which lets you keep settings and data in the same location as your installation, for example, on a USB drive.

Customize your keyboard shortcuts

Keyboard Shortcut: `Ctrl+K Ctrl+S`(按两次)

You can search for shortcuts and add your own keybindings to the `keybindings.json` file.

```
{
  "key":"cmd+y",
  "command": "redo",
  "when":"editorTextFocus"
}
```

See more in Key Bindings for Visual Studio Code.

Multi cursor selection

To add cursors at arbitrary positions, select a position with your mouse and use `Alt+Click` (`Option+click` on macOS).

To set cursors above or below the current position use `Ctrl+Alt+Up` or `Ctrl+Alt+Down`

You can add additional cursors to all occurrences of the current selection with `Ctrl+Shift+L.`

> Note: You can also change the modifier to `Ctrl/Cmd` for applying multiple cursors with the `editor.multiCursorModifier` setting

If you do not want to add all occurrences of the current selection, you can use `Ctrl+D` instead. This only selects the next occurrence after the one you selected so you can add selections one by one.

Column (box) selection

You can select blocks of text by holding `Shift+Alt` (`Shift+Option` on macOS) while you drag your mouse. A separate cursor will be added to the end of each selected line.

Git integration

Keyboard Shortcut: `Ctrl+Shift+G`

Diffs

From the **Source Control** view, select the file to diff. *Default is side by side diff*.

Inline view

Toggle inline view by clicking the **More Actions** (...) button in the top right and selecting **Switch to Inline View**. If you prefer the inline view, you can set `"diffEditor.renderSideBySide": false`.

Review pane

Navigate through diffs with `F7` and `Shift+F7`. This will present them in a unified patch format. Lines can be navigated with arrow keys and pressing `Enter` will jump back in the diff editor and the selected line.

Edit pending changes

You can make edits directly in the pending changes of the diff view.

Branches

Easily switch between Git branches via the Status Bar.(左下角)

Stage all

Hover over the number of files and click the plus button.(颜色会由灰**U**变为绿**A**)

Stage selected

Stage a portion of a file by selecting that file (using the arrows) and then choosing **Stage Selected Ranges** from the **Command Palette**.

Undo last commit

点右上角的**...**，再点击**Undo Last Commit**

See Git output

VS Code makes it easy to see what Git commands are actually running. This is helpful when learning Git or debugging a difficult source control issue.

Use the **Toggle Output** command (`Ctrl+Shift+U`) and select **Git** in the drop-down.

Resolve merge conflicts

During a merge, go to the **Source Control** view (`Ctrl+Shift+G`) and make changes in the diff view.

Set VS Code as default merge tool

```
git config --global merge.tool code
```

Debugging

Configure debugger

Open the **Command Palette** (`Ctrl+Shift+P`) and select **Debug**: Open `launch.json`, which will prompt you to select the environment that matches your project (Node.js, Python, C++, etc). This will generate a `launch.json` file. Node.js support is built-in and other environments require installing the appropriate language extensions. See the debugging documentation for more details.

Data inspection

Inspect variables in the **Debug** panels and in the console. (注意在dubug console输入变量名，类似在浏览器里dubug js)

Inline values

You can set `"debug.inlineValues": true` to see variable values inline in the debugger. This feature can be expensive and may slow down stepping, so it is disabled by default

Task runner

Auto detect tasks

Select **Terminal** from the top-level menu, run the command **Configure Tasks**, then select the type of task you'd like to run. This will generate a `tasks.json` file. See the Tasks documentation for more details.

There are occasionally issues with auto generation. Check out the documentation for getting things to work properly.

Run tasks from the Terminal menu

Select **Terminal** from the top-level menu, run the command **Run Task**, and select the task you want to run. Terminate the running task by running the command **Terminate Task**

Define keyboard shortcuts for tasks

You can define a keyboard shortcut for any task. From the **Command Palette** (`Ctrl+Shift+P`), select **Preferences: Open Keyboard Shortcuts File**, bind the desired shortcut to the `workbench.action.tasks.runTask` command, and define the Task as `args`.

For example, to bind `Ctrl+H` to the `Run tests` task, add the following:

```
{
  "key": "ctrl+h",
  "command": "workbench.action.tasks.runTask",
  "args": "Run tests"
}
```

Run npm scripts as tasks from the explorer

With the setting `npm.enableScriptExplorer`, you can enable an explorer that shows the scripts defined in your workspace.

我不知道npm是什么。。。

User Interface

Side by side editing

You can open as many editors as you like side by side vertically and horizontally. If you already have one editor open, there are multiple ways of opening another editor to the side of the existing one:

- `alt` click on a file in the Explorer.
- `ctrl+\` to split the active editor into two.
- **Open to the Side** (`Ctrl+Enter`) from the Explorer context menu on a file.
- Click the **Split** Editor button in the upper right of an editor.
- Drag and drop a file to any side of the editor region.
- `ctrl+Ente`r (macOS: `cmd+Enter`) in the **Quick Open** (`Ctrl+P`) file list.

Whenever you open another file, the editor that is active will display the content of that file. So if you have two editors side by side and you want to open file 'foo.cs' into the right-hand editor, make sure that editor is active (by clicking inside it) before opening file 'foo.cs'.

By default editors will open to the right-hand side of the active one. You can change this behavior through the setting `workbench.editor.openSideBySideDirection` and configure to open new editors to the bottom of the active one instead.

When you have more than one editor open you can switch between them quickly by holding the `ctrl`(macOS: `cmd`) key and pressing `1`, `2`, or `3`.

> Tip: You can resize editors and reorder them. Drag and drop the editor title area to reposition or resize the editor.

Minimap

You can click or drag the shaded area to quickly jump to different sections of your file. You can move the minimap to the left hand side or disable it completely by respectively setting `"editor.minimap.side": "left"` or `"editor.minimap.enabled": false` in your user or workspace settings.

Indent Guides

The image above also shows indentation guides (vertical lines) which help you quickly see matching indent levels. If you would like to disable indent guides, you can set `"editor.renderIndentGuides": false` in your user or workspace settings.

Breadcrumbs

The editor has a navigation bar above its contents called Breadcrumbs. It shows the current location and allows you to quickly navigate between folders, files, and symbols.

Breadcrumbs always show the file path and if the current file type has language support for symbols, the symbol path up to the cursor position. You can disable breadcrumbs with the **View > Show Breadcrumbs** toggle command. For more information about the breadcrumbs feature, such as how to customize their appearance, see the Breadcrumbs section of the Code Navigation article.

在tab和代码之间，似乎还挺有参考性的

Explorer

You can drag and drop files into the Explorer from outside VS Code to copy them (if the explorer is empty VS Code will open them instead)

VS Code works very well with other tools that you might use, especially command-line tools. If you want to run a command-line tool in the context of the folder you currently have open in VS Code, right-click the folder and select **Open in Command Prompt** (or **Open in Terminal** on macOS or Linux).

You can also navigate to the location of a file or folder in the native Explorer by right-clicking on a file or folder and selecting **Reveal in Explorer** (or **Reveal in Finder** on the macOS or **Open Containing Folder** on Linux).

> Tip: Type `Ctrl+P` (**Quick Open**) to quickly search and open a file by its name.

By default, VS Code excludes some folders from the Explorer (for example. `.git`). Use the `files.exclude` setting to configure rules for hiding files and folders from the Explorer.

> Tip: This is really useful to hide derived resources files, like `\*.meta` in Unity, or `\*.js` in a TypeScript project. For Unity to exclude the `\*.cs.meta` files, the pattern to choose would be: `"**/*.cs.meta": true`. For TypeScript, you can exclude generated JavaScript for TypeScript files with: `"**/*.js": {"when": "$(basename).ts"}`.

Multi-selection

You can select multiple files in the **File Explorer** and **OPEN EDITORS** view to run actions (Delete, Drag and Drop, Open to the Side) on multiple items. Use the `ctrl/cmd` key with `click` to select individual files and `shift` + `click` to select a range. If you select two items, you can now use the context menu **Compare Selected** command to quickly diff two files.

Note: In earlier VS Code releases, clicking with the `ctrl/cmd` key pressed would open a file in a new Editor Group to the side. If you would still like this behavior, you can use the `workbench.list.multiSelectModifier` setting to change multi-selection to use the Alt key.

```
"workbench.list.multiSelectModifier": "alt"
```

Filtering the document tree

You can type to filter the currently visible files in the **File Explorer**. With the focus on the **File Explorer** start to type part of the file name you want to match. You will see a filter box in the top-right of the **File Explorer** showing what you have typed so far and matching file names will be highlighted. When you press the cursor keys to move up and down the file list, it will jump between matching files or folders.

Hovering over the filter box and selecting **Enable Filter on Type** will show only matching files/folders. Use the 'X' **Clear** button to clear the filter.

Keyboard Shortcut

interface

`ctrl+shift+p`/`cmd+shift+p`: Command Palette. Type `?` to view help suggestions.

`alt`: Pressing the `alt` key enables fast scrolling in the editor and Explorers. By default, fast scrolling uses a 5X speed multiplier but you can control the multiplier with the **Editor: Fast Scroll Sensitivity** (`editor.fastScrollSensitivity`) setting.

`ctrl+P`: quick open

`` ctrl+` ``: integrated terminal

`ctrl+B`: toggle sidebar `ctrl+K Z`: zen mode. Press `Esc` twice to exit Zen Mode.

`ctrl+\`: side by side editing. You can also drag and drop editors to create new editor groups and move editors between groups. Switch between editors: `Ctrl+1`, `Ctrl+2`, `Ctrl+3`

`Ctrl+Shift+V`: Open Markdown preview. Use `Ctrl+K V` for side by side Markdown edit and preview.

explorer

`ctrl+Tab`: Navigate entire history. Navigate back: `Alt+Left`; Navigate forward: `Alt+Right`

`ctrl+click/cmd+click`: You can quickly open a file or image or create a new file by moving the cursor to the file link. (打开一个文件，多用于js，因为有), alternatively, you can select a symbol then type `F12`(go to definition)

Alternatively, you can use the context menu.

Select a symbol then type `alt+F12`: peek(我认为是看其他文件中所有使用这个symbol的地方)

Select a symbol then type `shift+F12`: go to reference

Select a symbol then type `shift+alt+F12` to open the References view showing all your file's symbols in a dedicated view.

You can go back to your previous location with the **Go > Back** command or `Alt+Left`.

You can also see the type definition if you press `Ctrl` (`Cmd` on macOS) when you are hovering over the type.

Select a symbol then type `F2`: rename a symbol

Search and modify

Besides searching and replacing expressions, you can also search and reuse parts of what was matched, using regular expressions with capturing groups. Enable regular expressions in the search box by clicking the **Use Regular Expression** `.*` button (`alt+R`) and then write a regular expression and use parenthesis to define groups. You can then reuse the content matched in each group by using `$1`, `$2`, etc. in the Replace field.

coding

`Shift+Alt+Up` or `Shift+Alt+Down`: Copy line up / down

> The commands **Copy Line Up/Down** are unbound on **Linux** because the VS Code default keybindings would conflict with Ubuntu keybindings, see Issue #509. You can still set the commands `editor.action.copyLinesUpAction` and `editor.action.copyLinesDownAction` to your own preferred keyboard shortcuts.

`Alt+Up` or `Alt+Down`: Move line up and down(把这行代码上移或下移)

`Shift+Alt+Left` or `Shift+Alt+Right`: Shrink / expand selection(从最外围`{}`的到最内的`{}`)

`Ctrl+Shift+[` and `Ctrl+Shift+]`: Code folding(打开或是关闭里`{}`的代码)

`Ctrl+K Ctrl+F`: Code formatting. Whole document format: `Shift+Alt+F`(自动对齐)

`Ctrl+K Ctrl+X`: Trim trailing whitespace(去掉多余的空格)

`Ctrl+U`: Undo cursor position

`Ctrl+L`: Select current line

`Ctrl+G`: Navigate to a specific line

`Ctrl+Home` and `Ctrl+End`: Navigate to beginning and end of file

`Ctrl+Shift+O`: Go to Symbol in File. You can group the symbols by kind by adding a colon, `@:`.

`Ctrl+T`: Go to Symbol in Workspace

`Ctrl+Space`: trigger the Suggestions widget by IntelliSense.