重慶理工大學

VSCode 使用手册 V1.0

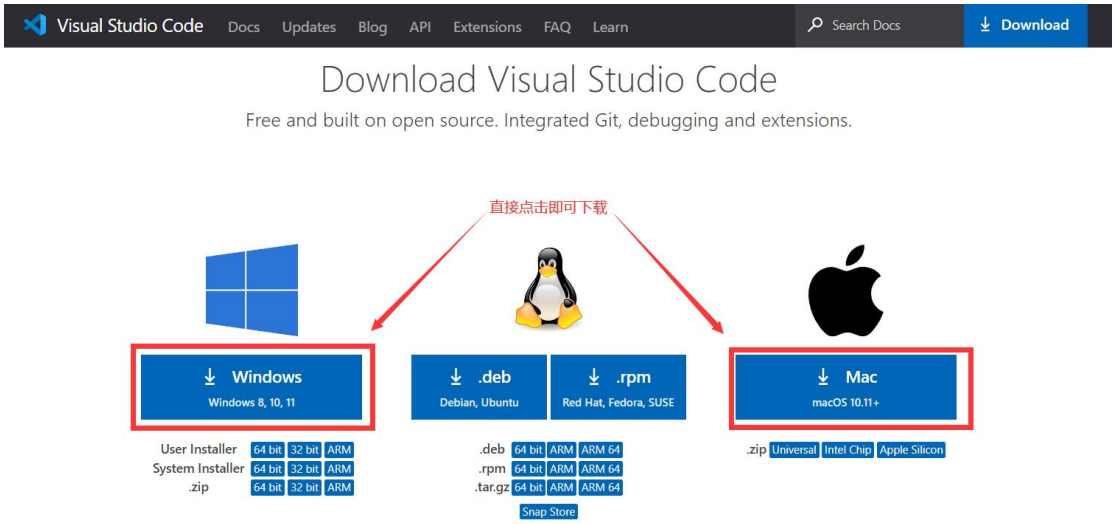
2022 年 9 月

目录

1. VSCode 的下载安装及插件

1.1 VSCode 的下载

VSCode 官网下载地址: https://code.visualstudio.com/download



但是由于是使用国内的镜像地址下载会导致下载速度极慢，因此可以使用以下两个下载链接

Window 端 x64-1.71.0 版:

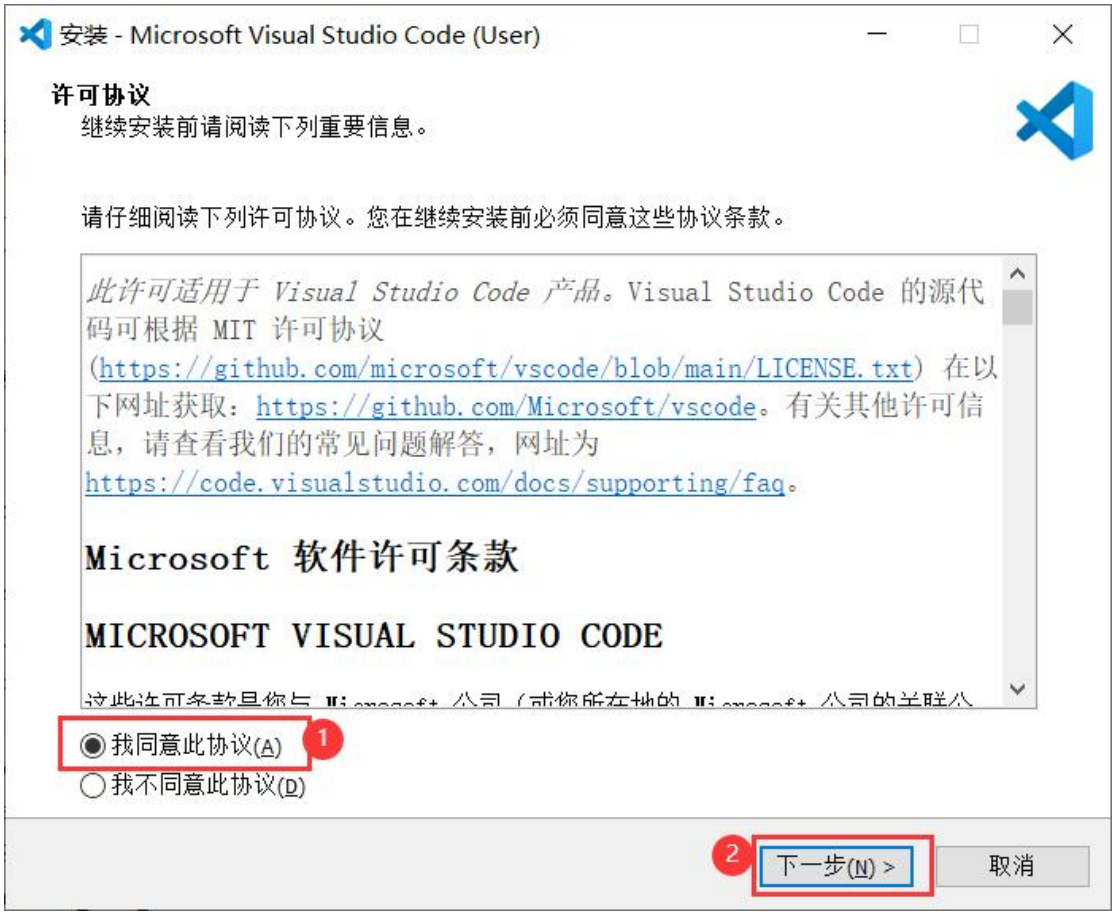
https://vscode.cdn.azure.cn/stable/784b0177c56c607789f9638da7b6bf3230d47a8c/VSCodeUserSetup-x64-1.71.0.exe

Mac 端:

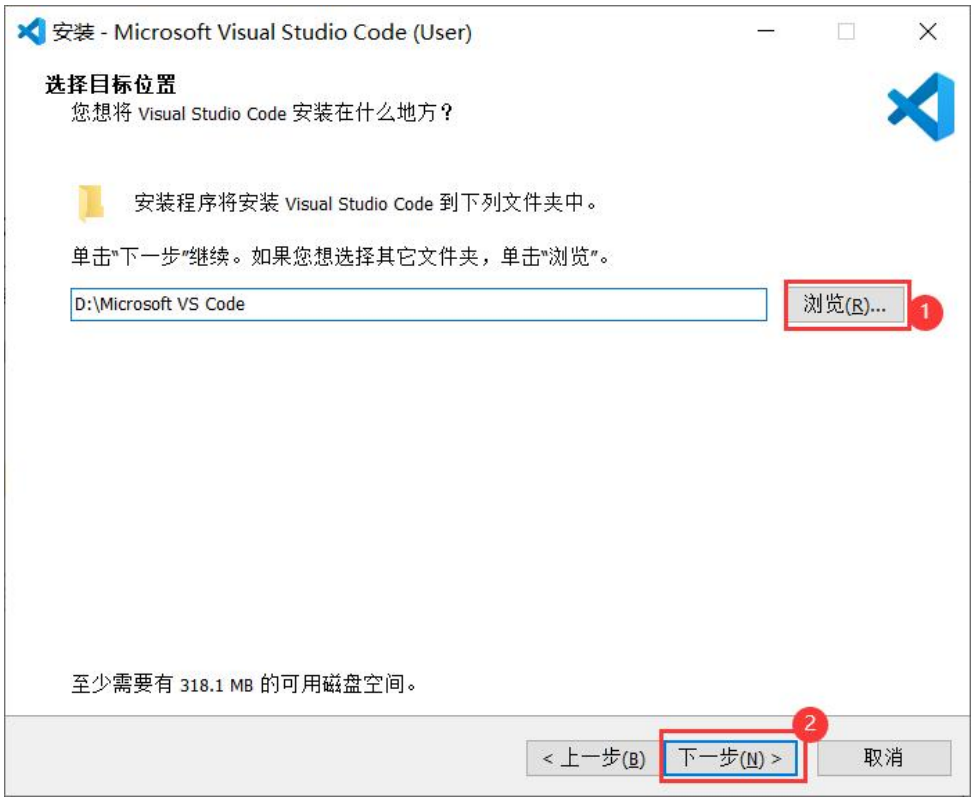
https://vscode.cdn.azure.cn/stable/784b0177c56c607789f9638da7b6bf3230d47a8c/VSCode-darwin-universal.zip

1.2 VSCode 的安装

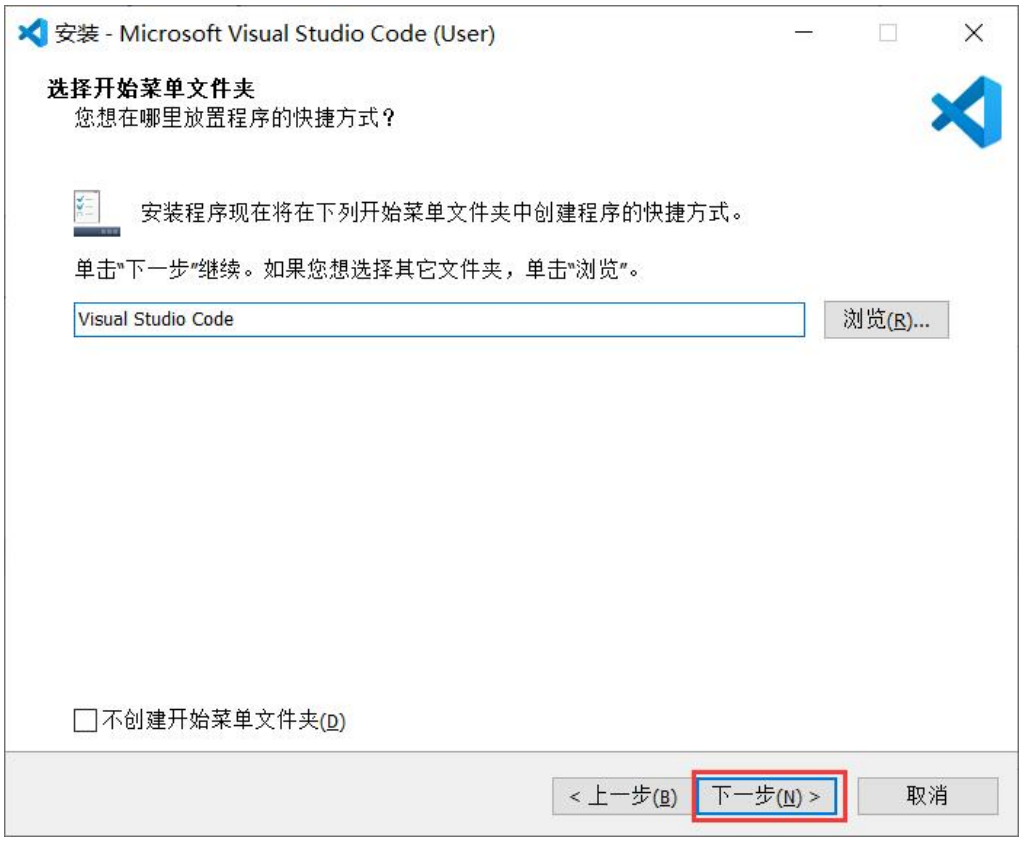
1>下载完成后，双击打开应用程序



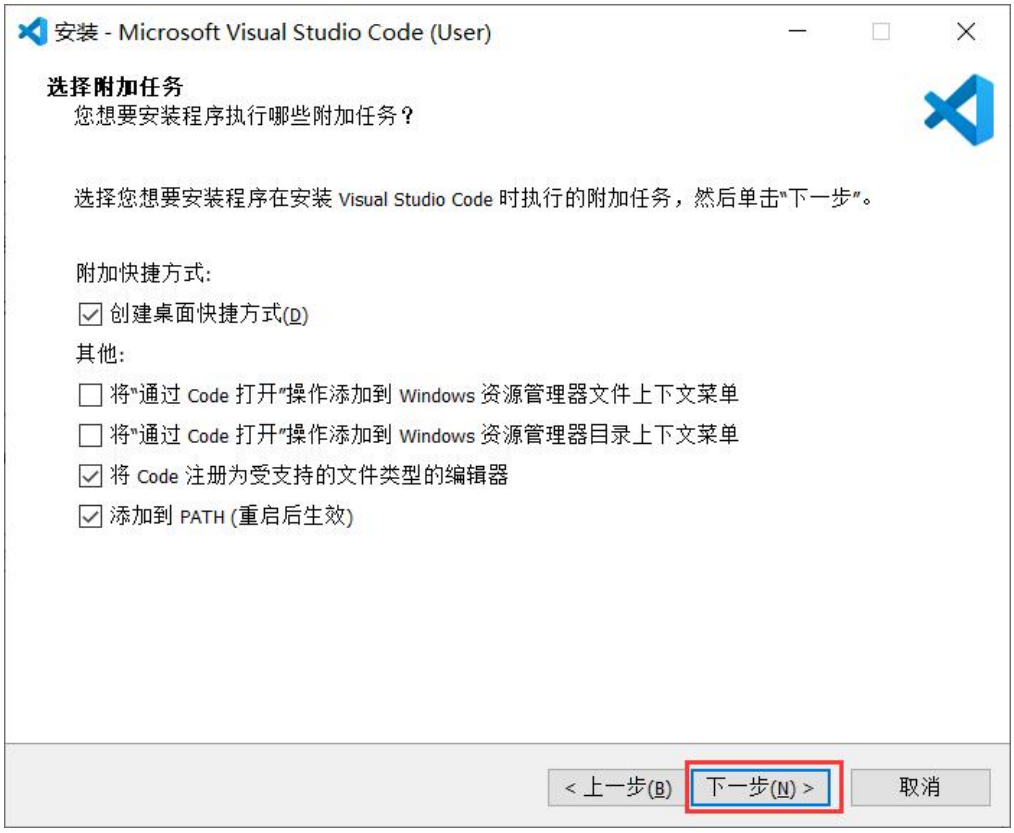
2>选择软件安装路径并点击下一步



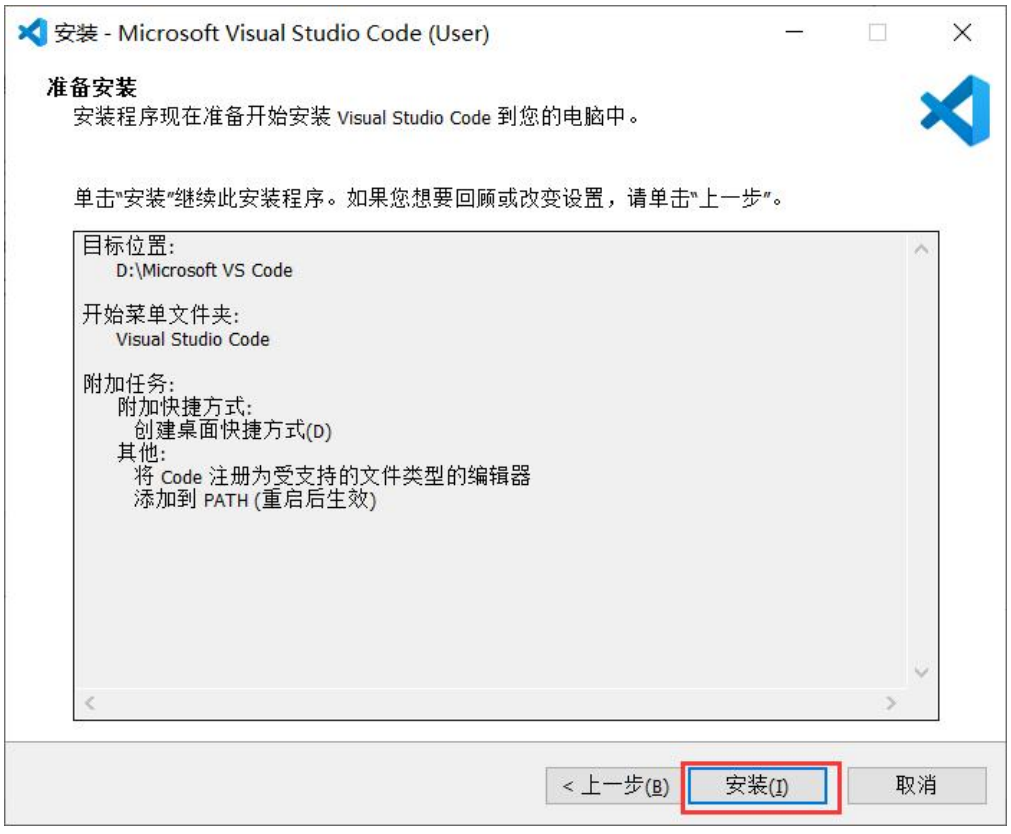
3>直接点击下一步



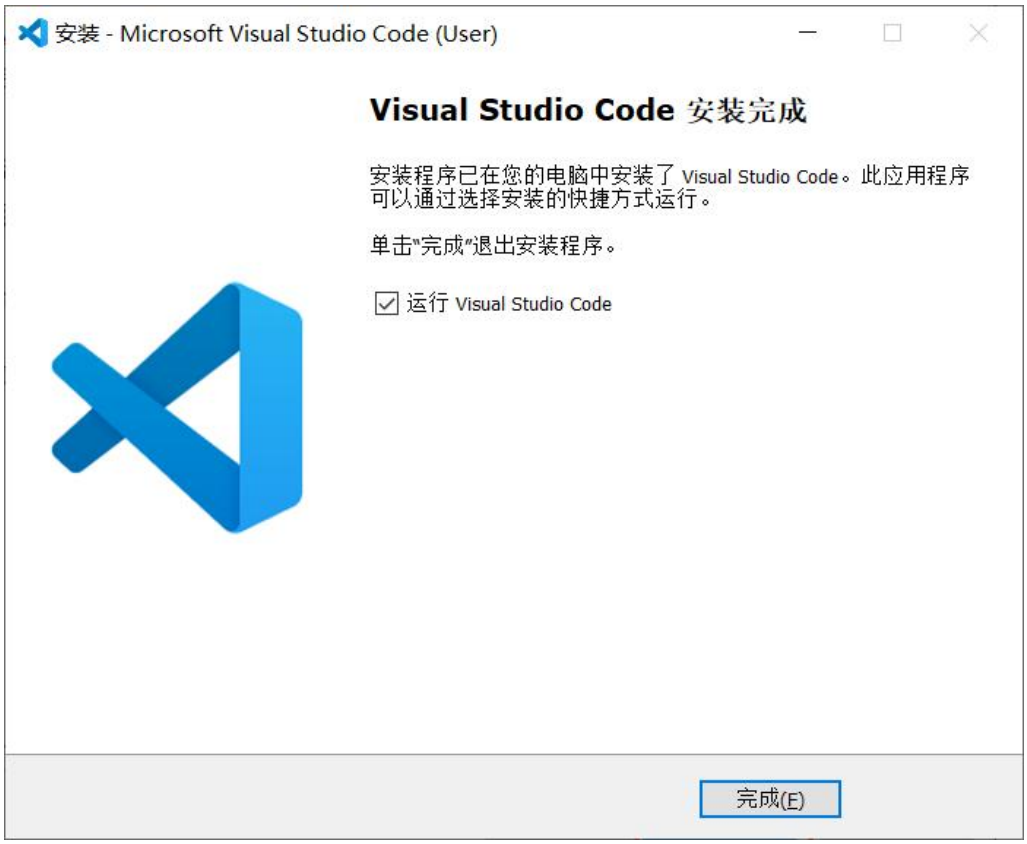
4>选择要安装的附加任务并点击下一步



5>点击安装

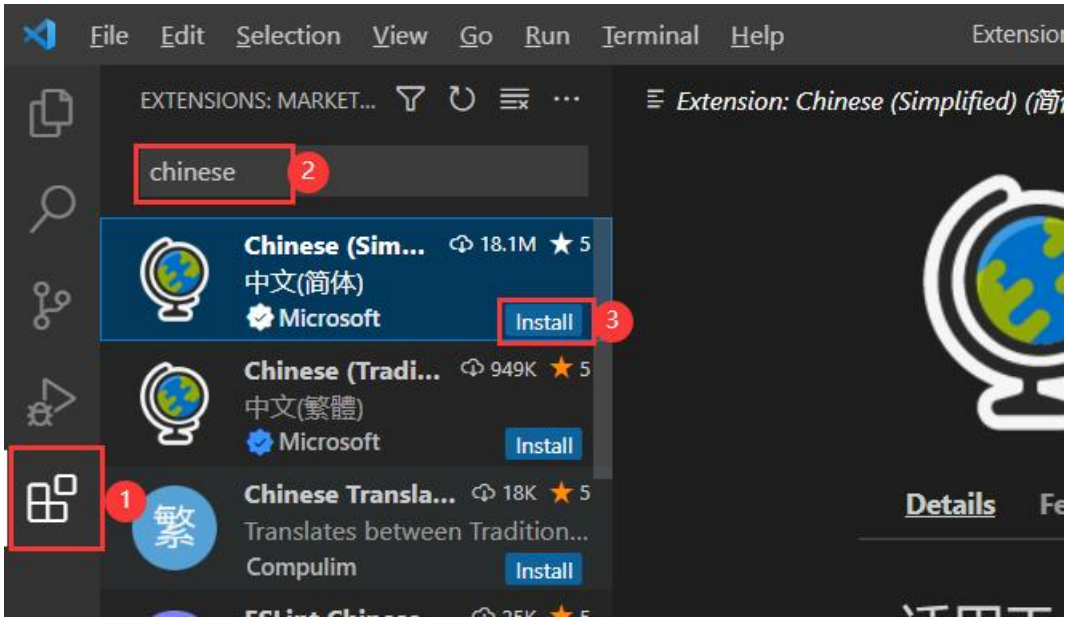


6>等待，安装完毕

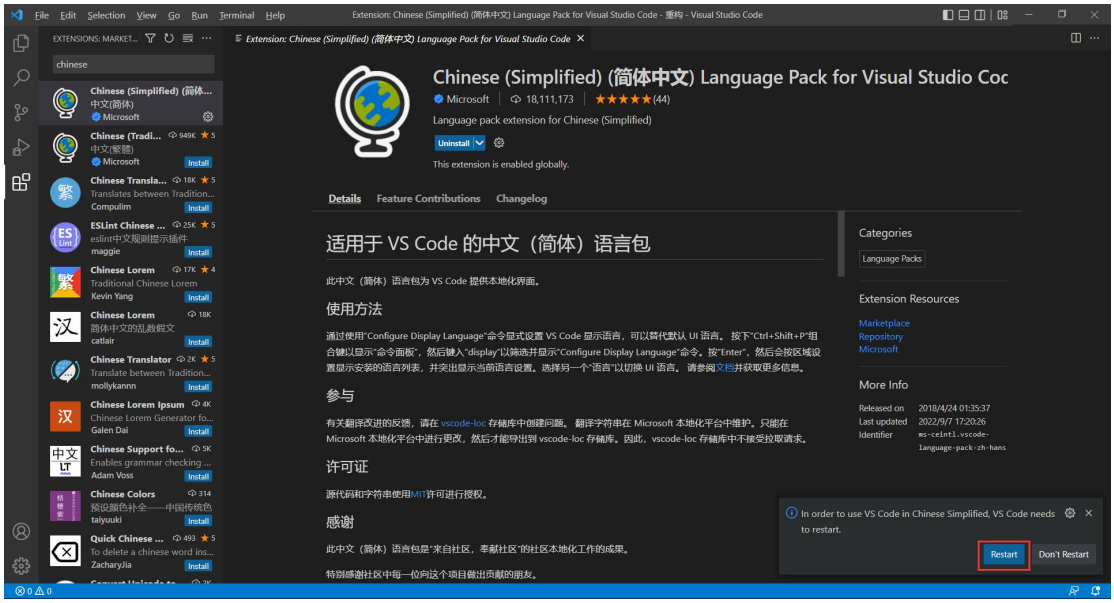


1.3 安装中文插件

1> 打开 VSCode 软件，按如下步骤下载中文插件



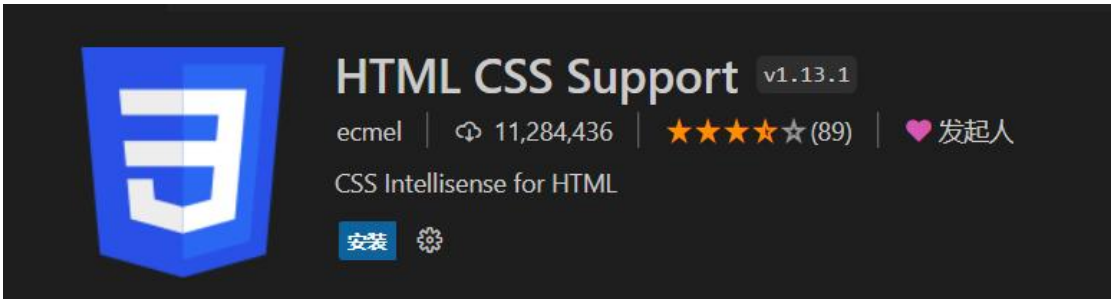
2> 重启软件



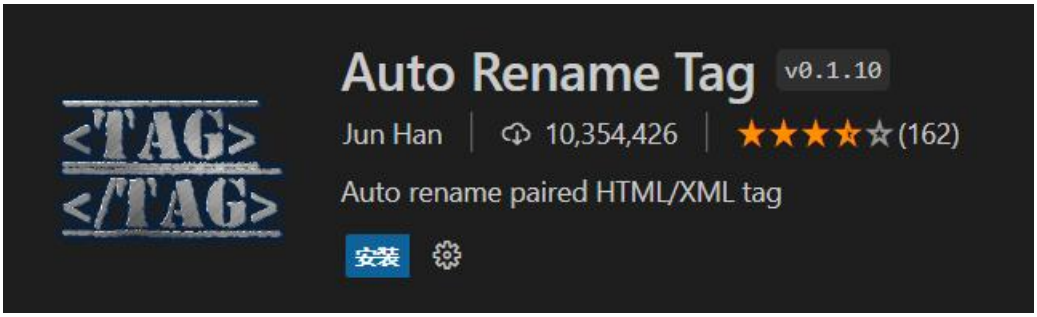
3> 再次进入软件后，就是中文界面了

1.4 其他推荐插件

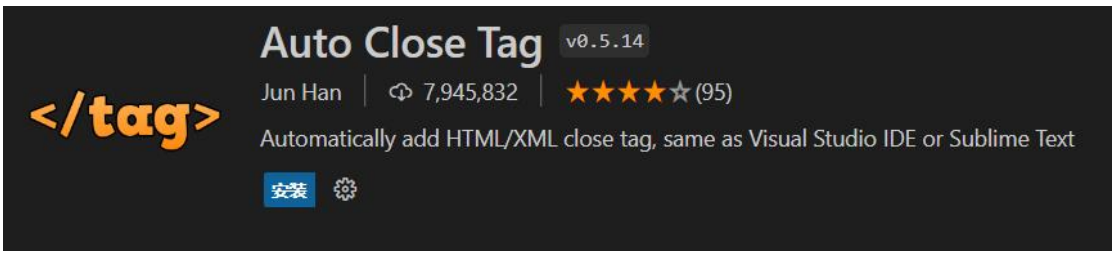
- **CSS 类名智能提示（HTML CSS Support）**：它会提示一些类名供你选择



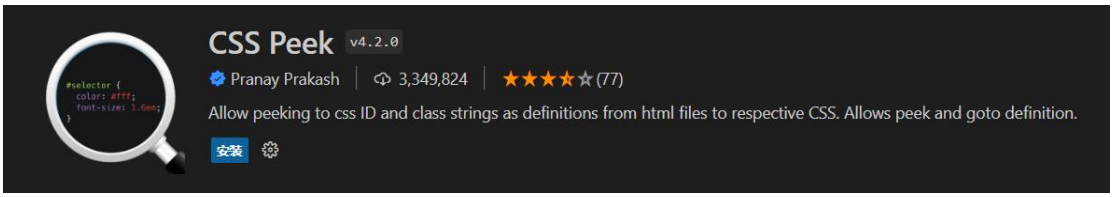
- **自动重命名标签（Auto Rename Tag）**：这个插件对你的标签修改起来一个很大的作用，当你修改起始标签的时候，结束标签也会随着起始标签的修改而修改



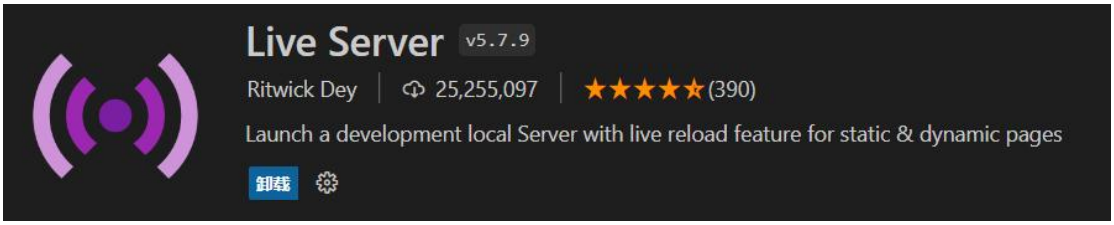
- **自动闭合标签（Auto Close Tag）：**安装了这个插件后，如果你的起始标签不小心删除的结束标签，只要打 </ 就会自动补全，根据就近原则，一次只能补全一个标签。



- **html 与 css 关联（CSS Peek）**：可以直接在 html 代码中，按 ctrl+鼠标左键查看该元素的样式。



- **自动刷新页面（Live Server）:**在本地启动一个服务器，代码写完后可以实现「热更新」，实时地在网页中看到运行效果。就不需要每次都得手动刷新页面了。使用方式：右键选择「Open with Live Server」



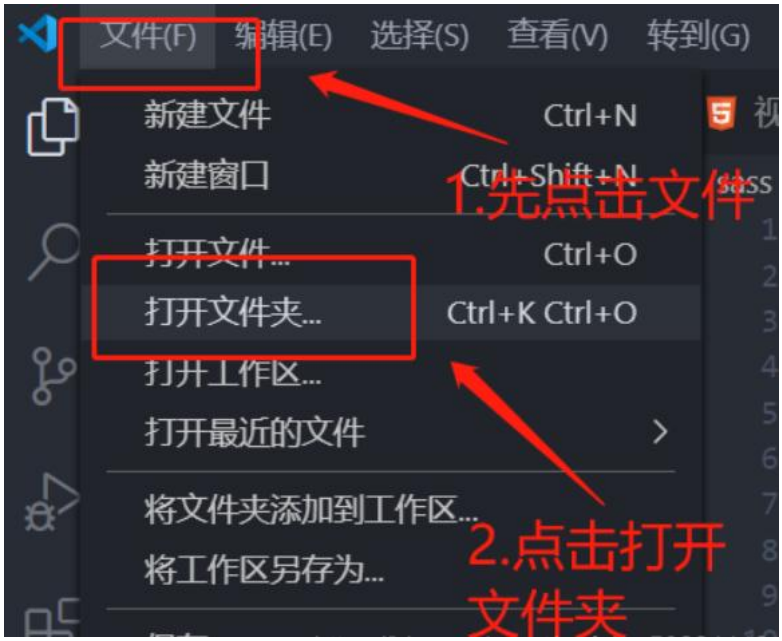
2. VSCode 的使用

2.1 VSCode 界面介绍

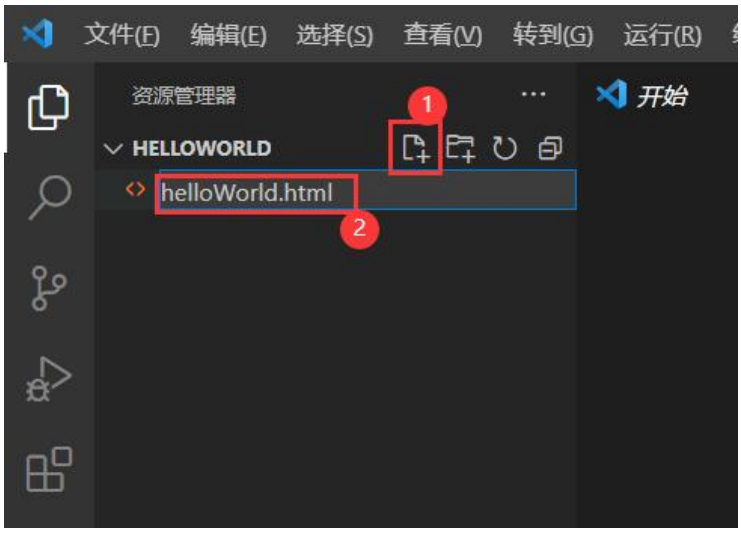
2.2 Hello World

1>创建一个空文件夹，作为项目目录

2>在文件中选择【打开文件夹】，当然，我们也可以直接将文件夹拖进 VSCode 编辑器

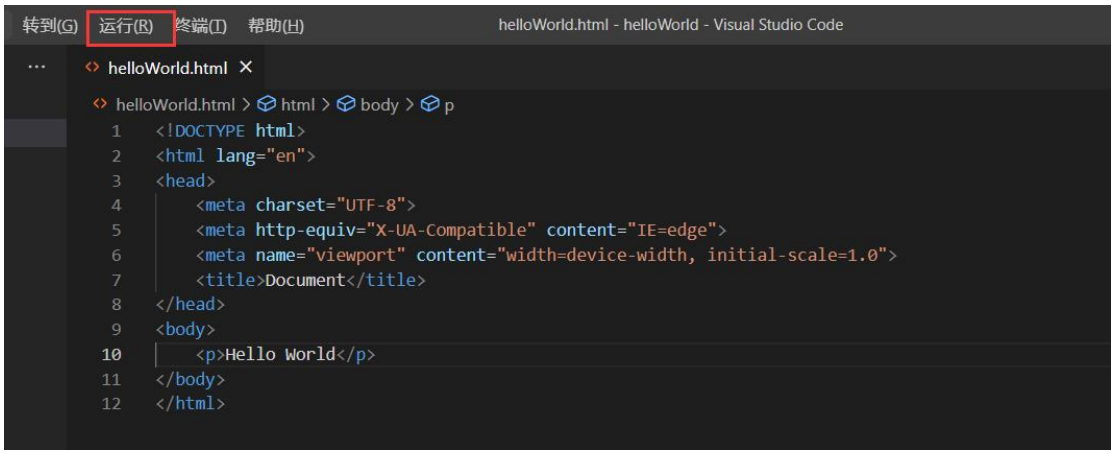


3>在左侧的“资源管理器”能看见我们的项目文件夹，右键它，新建一个 html 文件

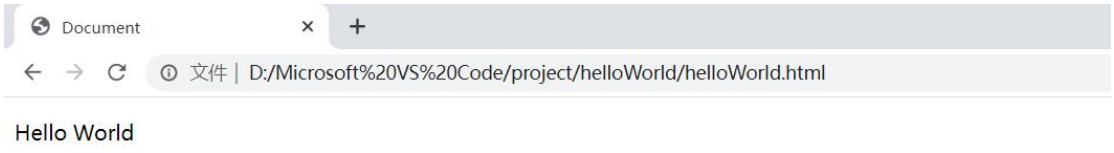


4> 输入半角感叹号!（英文感叹号）,按下 tab，将自动生成一个 html 结构

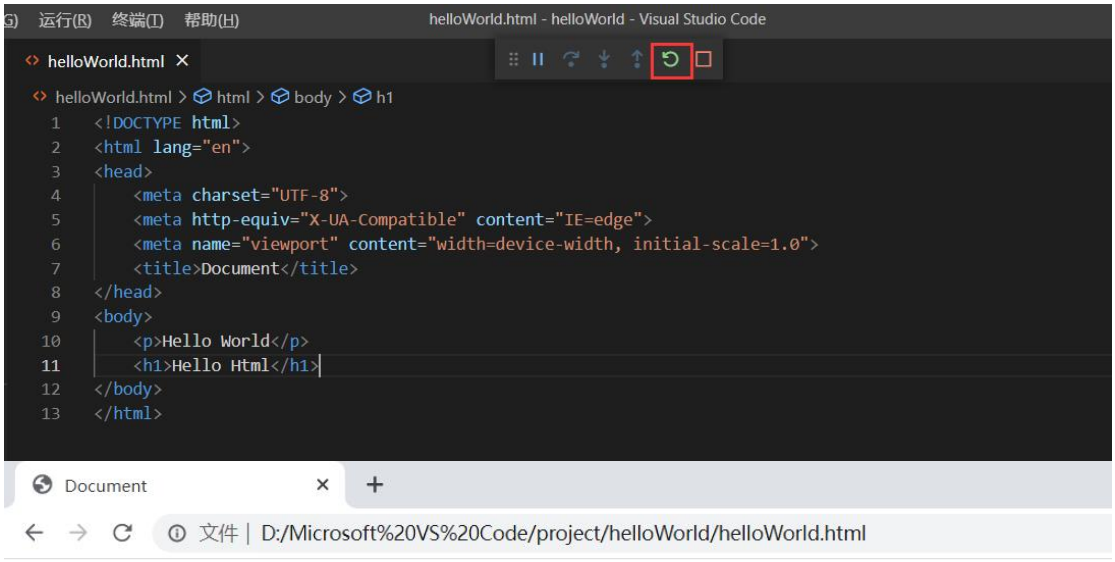
5> 在 body 里编写要显示的代码内容并保存文件，再点击上方导航栏的运行按钮，在弹出的菜单窗口中单击启动调试（或直接 F5 运行调试）



6> 在上方弹出的搜索框选择要运行的浏览器后将在浏览器上显示内容



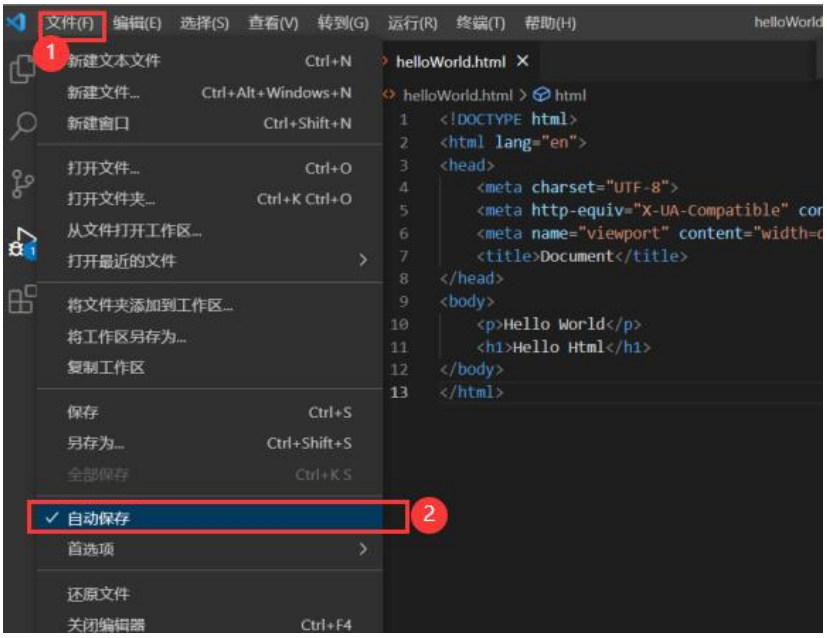
7> 若要对原来的代码进行增删改，仍然需要保存。可以通过单击 vscode 上方的刷新图标或浏览器内的刷新图标来直接刷新浏览器页面更新内容



2.3 可能有用的操作

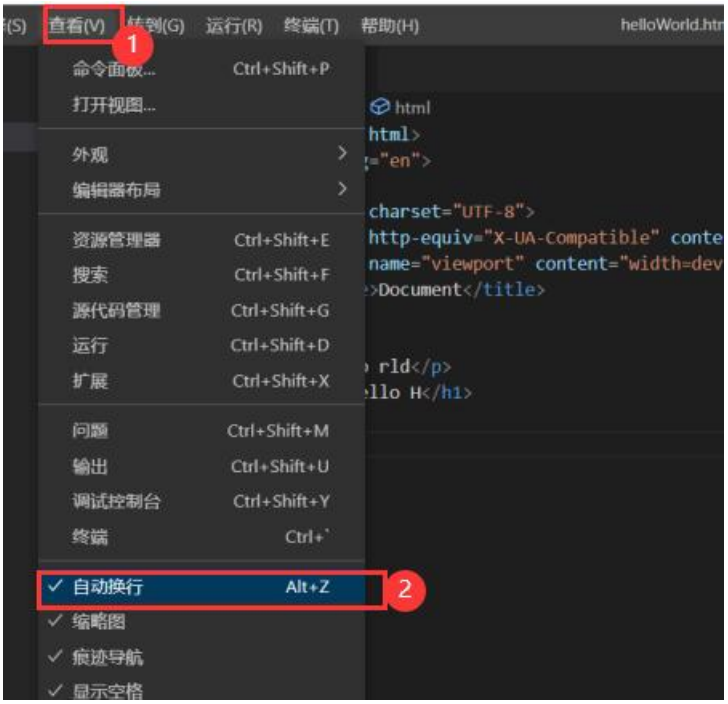
2.3.1 自动保存

修改完代码后，默认不会自动保存。使用自动保存更便捷



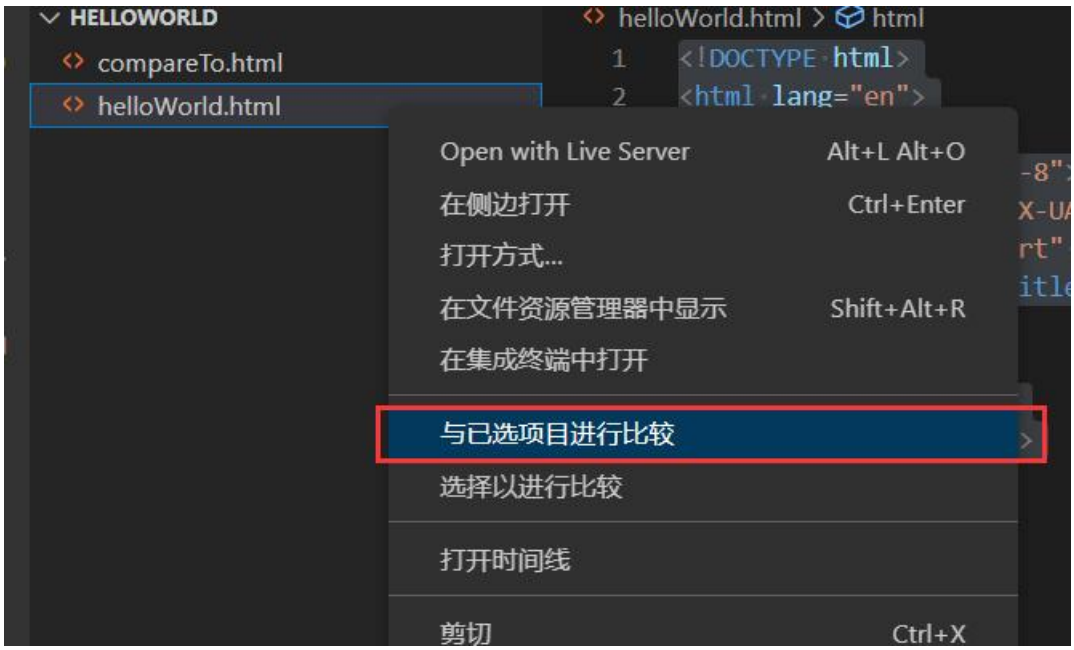
2.3.2 自动换行

当一行文字超出屏幕显示区，使用自动换行，能使代码显示更美观



2.3.3 文件对比

VSCode 默认支持两个文件进行内容对比，将一个文件右键选择以进行对比，另一文件右键选择与已选项目进行对比即可进行代码对比



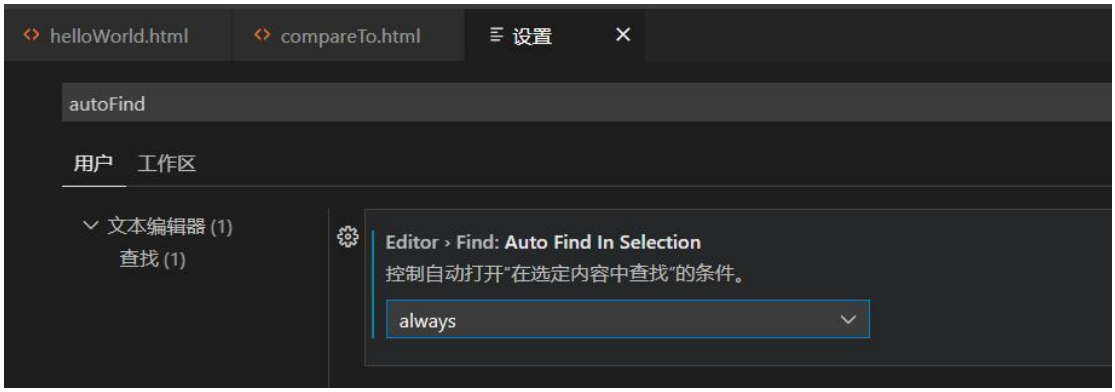
2.3.4 在当前文件中搜索

- Ctrl+F（mac 端是 Cmd+F）：在当前文件中搜索，光标在提示框内
- F3（mac 端是 Cmd+G）：在当前文件中搜索，光标仍停留在编辑器里

弹出的搜索框不同按钮有不同的功能

你可以通过「Tab」键和「Shift + Tab」键在输入框和替换框之间进行切换

「在选定内容中查找」这个功能还是比较实用的。你也可以在设置项里搜索 editor.find.autoFindInSelection，勾选该设置项后，那么，当你选中指定内容后，然后按住「Ctrl + F」，就可以**自动**只在这些内容里进行查找。该设置项如下图所示：



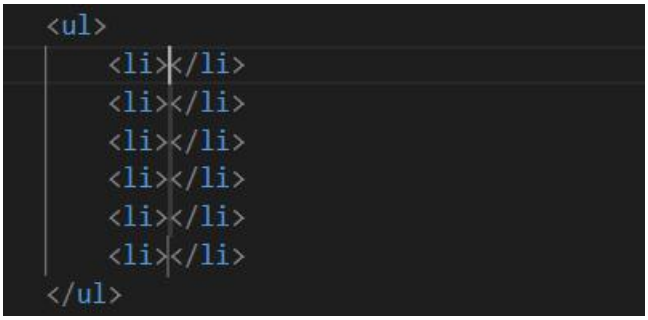
2.3.5 自带终端

- Ctrl+`：打开 VSCode 自带终端

2.3.6 Emmet

Emmet 可以极大的提高 html 和 css 的编写效率，它提供了一种非常简练的语法规则。

举个例子，我们在编辑器中输入缩写代码：ul>li*6，然后按下 Tab 键，即可得到如下代码片段：



2.3.7 工作区快捷键

Mac 快捷键	Win 快捷键	作用	备注
Cmd + Shift + P	**Ctrl + Shift + P**，F1	显示命令面板	
Cmd + B	**Ctrl + B**	显示/隐藏侧边栏	很实用
Cmd + \	Ctrl + \	**创建多个编辑器**	【重要】抄代码利器
Cmd + 1、2	**Ctrl + 1、2**	聚焦到第 1、第 2 个编辑器	同上重要
cmd +/-	**ctrl +/-**	将工作区放大/缩小（包括代码字体、左侧导航栏）	在投影仪场景经常用到
Cmd + J	Ctrl + J	显示/隐藏控制台	
Cmd + Shift + N	**Ctrl + Shift + N**	重新开一个软件的窗口	很常用
Cmd + Shift + W	Ctrl + Shift + W	关闭软件的当前窗口	
Cmd + N	Ctrl + N	新建文件	
Cmd + W	Ctrl + W	关闭当前文件	

2.3.8 跳转操作

Mac 快捷键	Win 快捷键	作用	备注
Cmd + `	没有	在同一个软件的**多个工作区**之间切换	使用很频繁
Cmd + Option + 左右方向键	Ctrl + Pagedown/Page up	在已经打开的**多个文件**之间进行切换	非常实用
Ctrl + Tab	Ctrl + Tab	在已经打开的多个文件之间进行跳转	不如上面的快捷键快
Cmd + Shift + O	Ctrl + shift + O	在当前文件的各种**方法之间**进行跳转	
Ctrl + G	Ctrl + G	跳转到指定行	
Cmd+Shift+\	Ctrl+Shift+\	跳转到匹配的括号	

2.3.9 移动光标

Mac 快捷键	Win 快捷键	作用	备注
方向键	方向键	在**单个字符**之间移动光标	大家都知道
option + 左右方向键	**Ctrl + 左右方向键**	在**单词**之间移动光标	很常用
Cmd + 左右方向键	**Fn + 左右方向键**	在**整行**之间移动光标	很常用
Cmd + ←	Fn + ←（或 Win + ←）	将光标定位到当前行的最左侧	很常用
Cmd + →	Fn + →（或 Win + →）	将光标定位到当前行的最右侧	很常用
Cmd + ↑	Ctrl + Home	将光标定位到文章的第一行	
Cmd + ↓	Ctrl + End	将光标定位到文章的最后一行	
Cmd + Shift + \		在**代码块**之间移动光标	

2.3.10 编辑操作

Mac 快捷键	Win 快捷键	作用	备注
Cmd + Enter	**Ctrl + Enter**	在当前行的下方新增一行，然后跳至该行	即使光标不在行尾，也能快速向下插入一行
Cmd+Shift+Enter	Ctrl+Shift+Enter	在当前行的上方新增一行，然后跳至该行	即使光标不在行尾，也能快速向上插入一行
Option + ↑	**Alt + ↑**	将代码向上移动	很常用
Option + ↓	**Alt + ↓**	将代码向下移动	很常用
Option + Shift + ↑	Alt + Shift + ↑	将代码向上复制	
Option + Shift + ↓	**Alt + Shift + ↓**	将代码向下复制	写重复代码的利器

2.3.11 多光标编辑

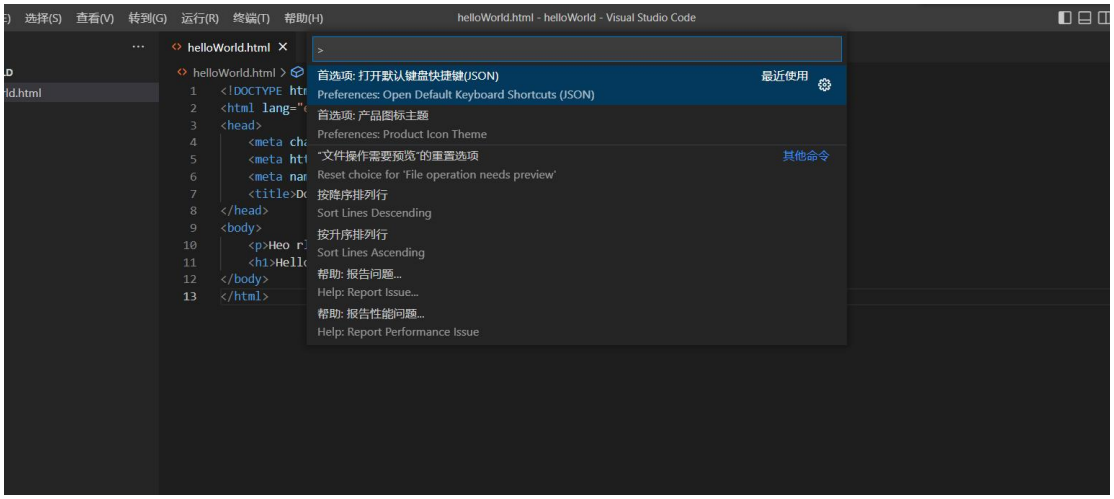
Mac 快捷键	Win 快捷键	作用	备注
Cmd + Option + 上下键	**Ctrl + Alt + 上下键**	在连续的多列上，同时出现光标	
Option + 鼠标点击任意位置	**Alt + 鼠标点击任意位置**	在任意位置，同时出现光标	
Option + Shift + 鼠标拖动	Alt + Shift + 鼠标拖动	在选中区域的每一行末尾，出现光标	
Cmd + Shift + L	Ctrl + Shift + L	在选中文本的所有相同内容处，出现光标	

2.3.12 删除操作

Mac 快捷键	Win 快捷键	作用	备注
Cmd + shift + K	Ctrl + Shift + K	删除整行	「Cmd + X」的作用是剪切，但也可以删除整行
option + Backspace	**Ctrl + Backspace**	删除光标之前的一个单词	英文有效，很常用
option + delete	Ctrl + delete	删除光标之后的一个单词	
Cmd + Backspace		删除光标之前的整行内容	很常用
Cmd + delete		删除光标之后的整行内容	

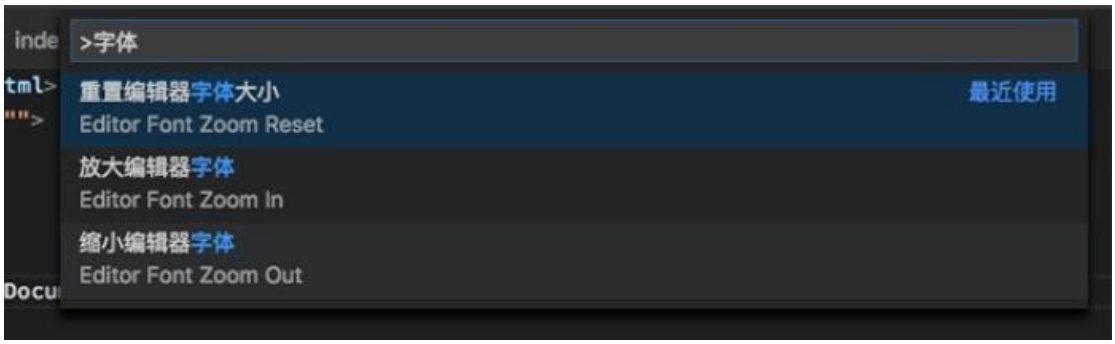
2.4 命令面板的使用

Windows 用户按住快捷键 Ctrl+Shift+P （Mac 用户按住快捷键 Cmd+Shift+P），可以打开命令面板。如图：



2.4.1 设置字体大小

在命令面板输入“字体”，可以进行字体的设置。如图：



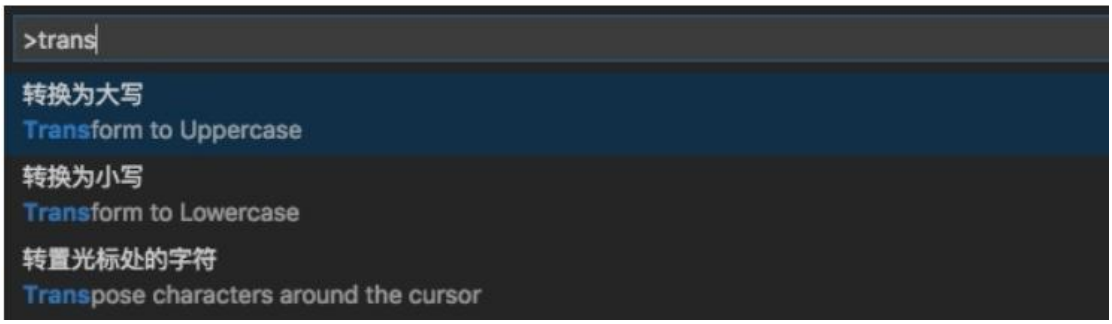
当然，你也可以在菜单栏，选择「首选项-设置-常用设置」，在这个设置项里修改字体大小。

2.4.2 自定义快捷键

在命令面板输入“快捷键”，就可以进入快捷键的设置。

2.4.3 大小写转换

选中文本后，在命令面板中输入 transfrom，就可以修改文本的大小写了。如图：



3.浏览器工具及调试

在 JavaScript 开发过程中，代码可能存在一些语法或者逻辑上的错误，导致程序不能得到我们想要的结果，这时就需要我们找到并修复这些错误，我们将查找和修复错误的过程称为调试或代码调试。

调试是程序开发过程中必不可少的一个环节，熟练掌握各种调试技巧，能在我们的工作中起到事半功倍的效果。

在前端开发中，想要快速定位错误，可以借助浏览器内置的调试工具（控制台），借助调试工具我们不仅可以很轻松的找到代码出错的位置，还可以通过设置断点（代码执行到断点处会暂停），来检查代码执行过程中变量的变化。

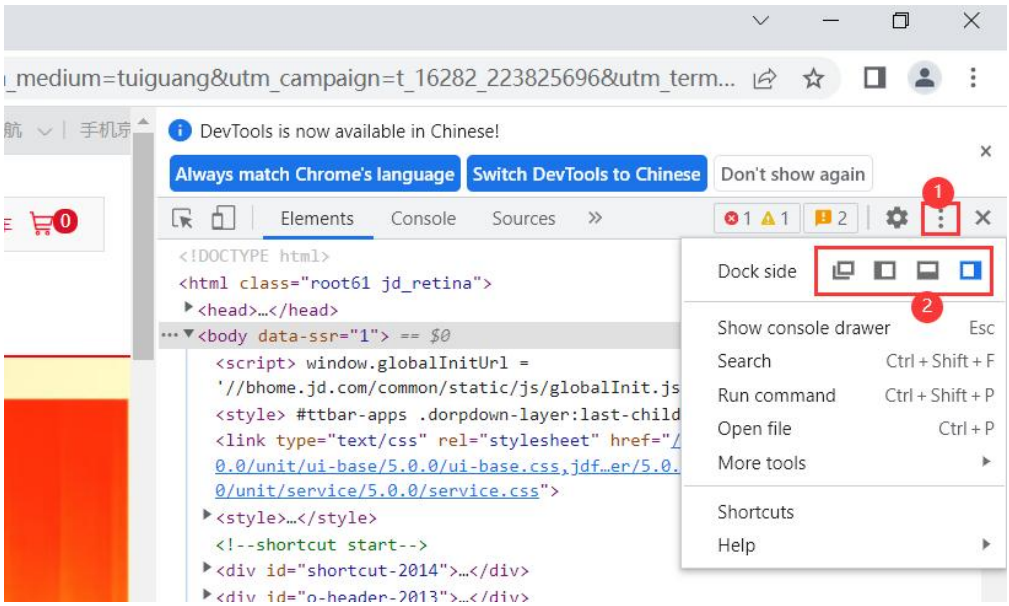
3.1 开发者模式

打开浏览器的开发者模式可以帮助我们更清晰的了解页面布局代码等控制台的相关信息

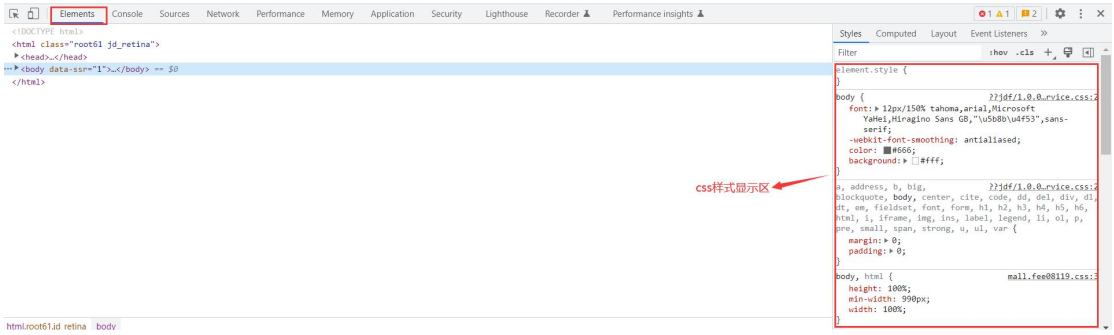
打开方式：

- 浏览器页面右键在弹出的菜单中点击检查
- 浏览器页面 F12
- 浏览器页面 Ctrl+Shift+i

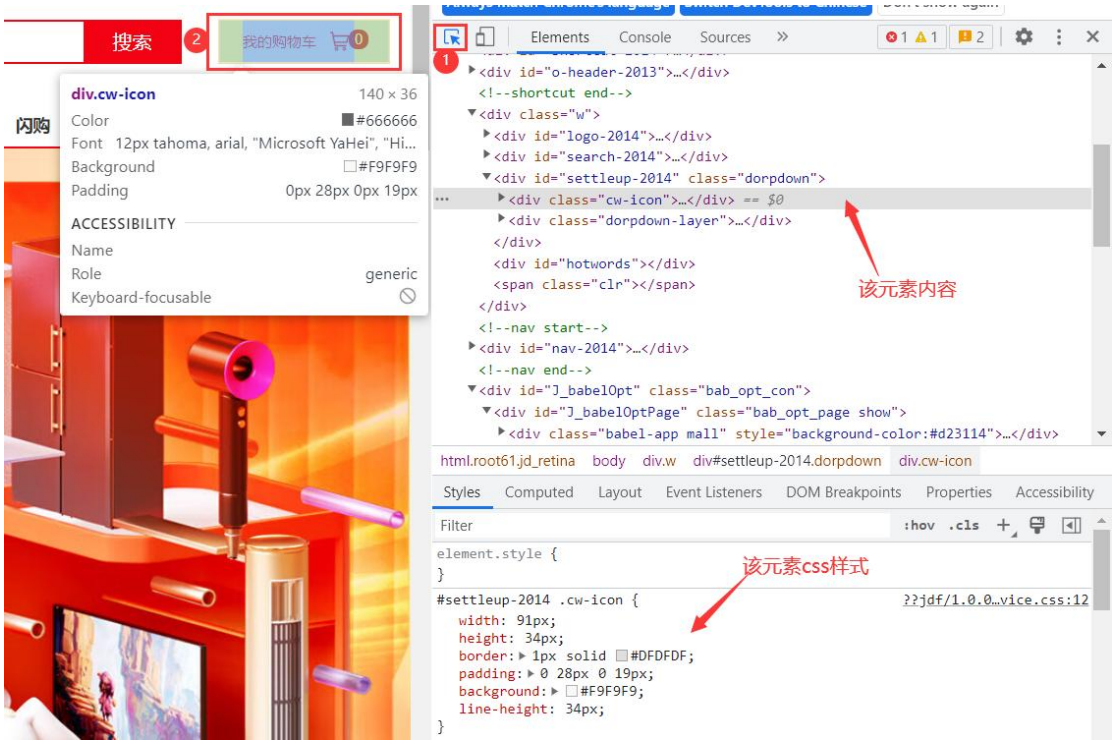
可以在 Dock side 处切换调试工具的显示位置



Elements 区可以看到整个页面的 html 元素，右侧的 Styles 区显示选中元素的样式



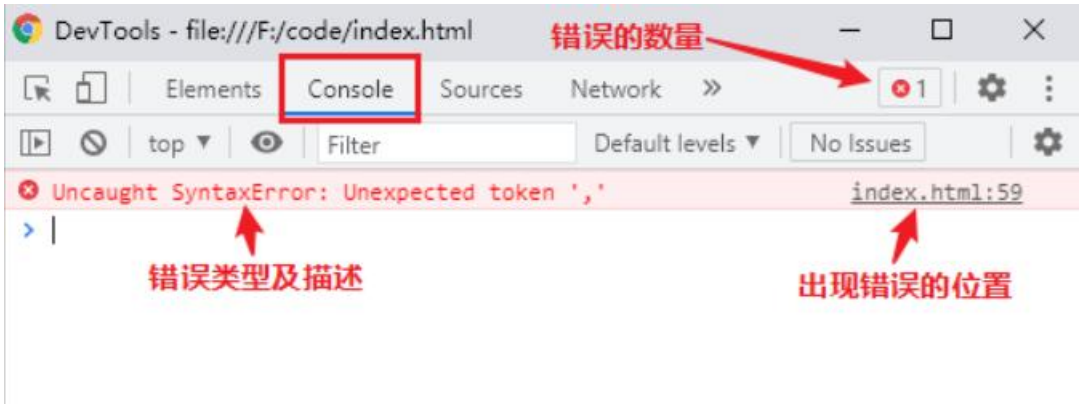
点击 1 处按钮后，将鼠标指针移至页面的任意 html 元素处，将在 Elements 区精确显示该元素的内容。点击后，Styles 区显示该元素 css 样式。例如：



可以直接在 Elements 区和 Styles 区修改代码，页面会随着更新内容，但修改的内容不会保存，即源代码不会改变。

3.2 控制台

控制台中能够显示代码中的语法错误和运行时错误，其中包括错误类型、错误描述以及错误出现的位置（即错误所在的行），如下图所示：



借助控制台提供的信息，我们可以轻松的定位代码中的错误，不过有一点需要注意，就是控制台提供的错误信息不一定百分之百正确，因为某些错误可能是

由于另外一个错误直接或间接引起的，所以控制台提示有错误的地方不一定真有问题。

3.3 如何调试 js 代码

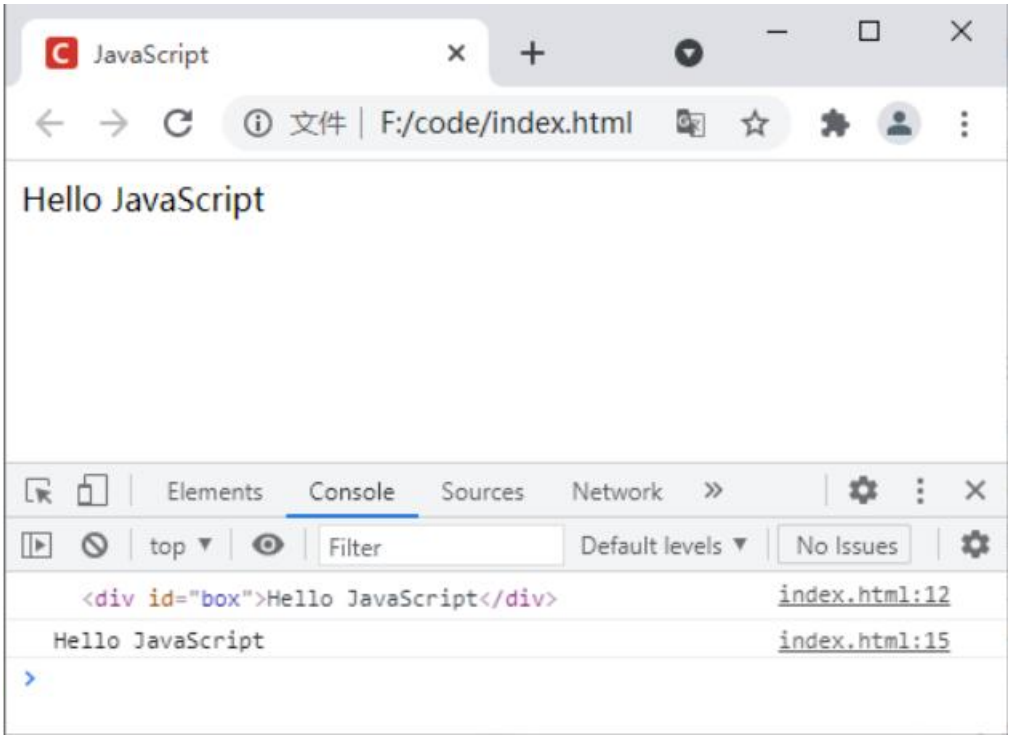
3.3.1 打印输出

有多种方法可以调试 JavaScript 代码，最简单的方法就是使用 console.log()、document.write()、alert() 等方法来打印程序中各个变量、对象、表达式的值，以确保程序每个阶段的运行结果都是正确的，推荐使用 console.log()，如下例所示

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript</title>
</head>

<body>
    <div id="box"></div>
    <script>
        var box = document.getElementById('box');
        console.log(box);
        var a = 'Hello', b = 'JavaScript';
        var c = a + ' ' + b;
        console.log(c);
        box.innerHTML = c;
    </script>
</body>
</html>
```

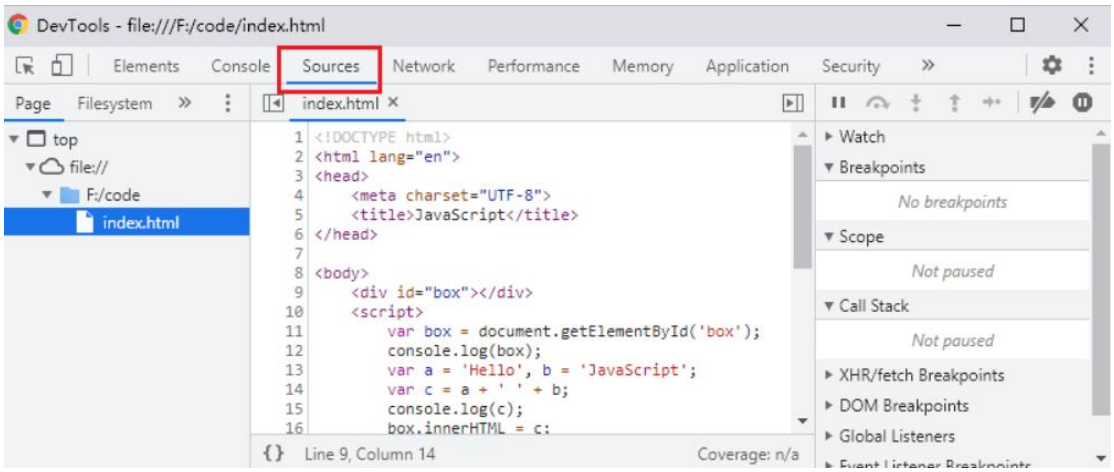
运行结果在控制台输出了 box 所在的标签和 c 值：



使用这种方法调试代码有一个弊端，那就是这些输出语句并不是代码中需要的，虽然它们不会影响代码的运行，但是为了代码更加整洁，在调试完程序后我们需要手动清理干净。

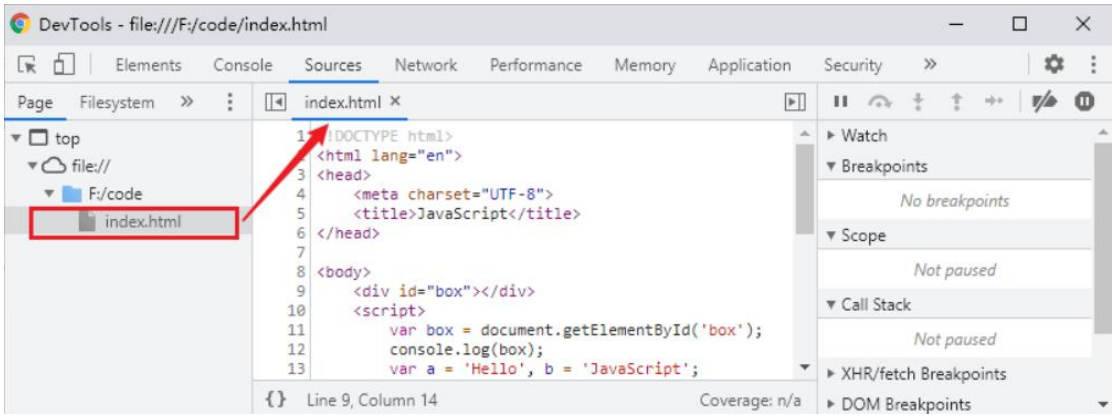
3.3.2 断点调试

断点是浏览器内置调试工具的重要功能之一，通过设置断点可以让程序在我们需要的地方中断（暂停），从而方便我们对该处代码进行分析和逻辑处理。以 Chrome 浏览器为例，找到调试工具中的 Sources 区：



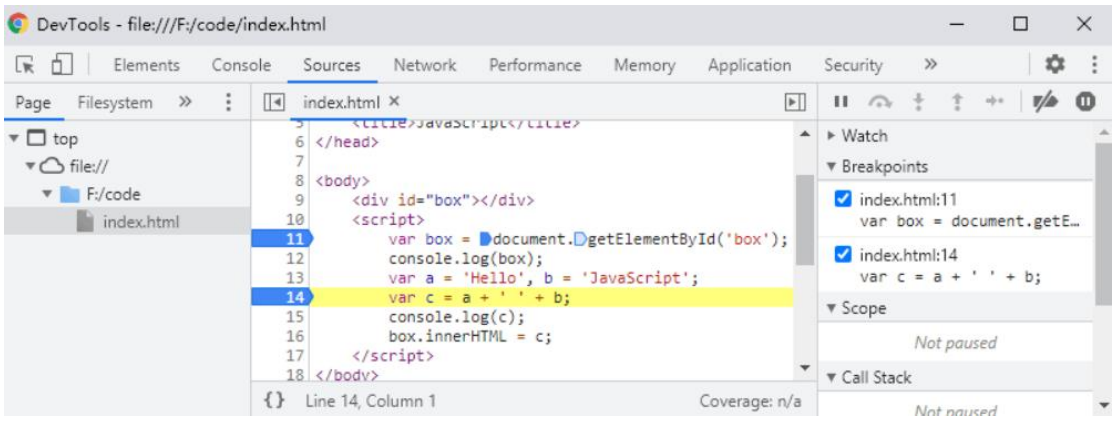
1>找到要调试的文件

打开调试工具后，需要在工具的左侧 Page 找到要调试的文件并单击打开该文件，如下图所示：



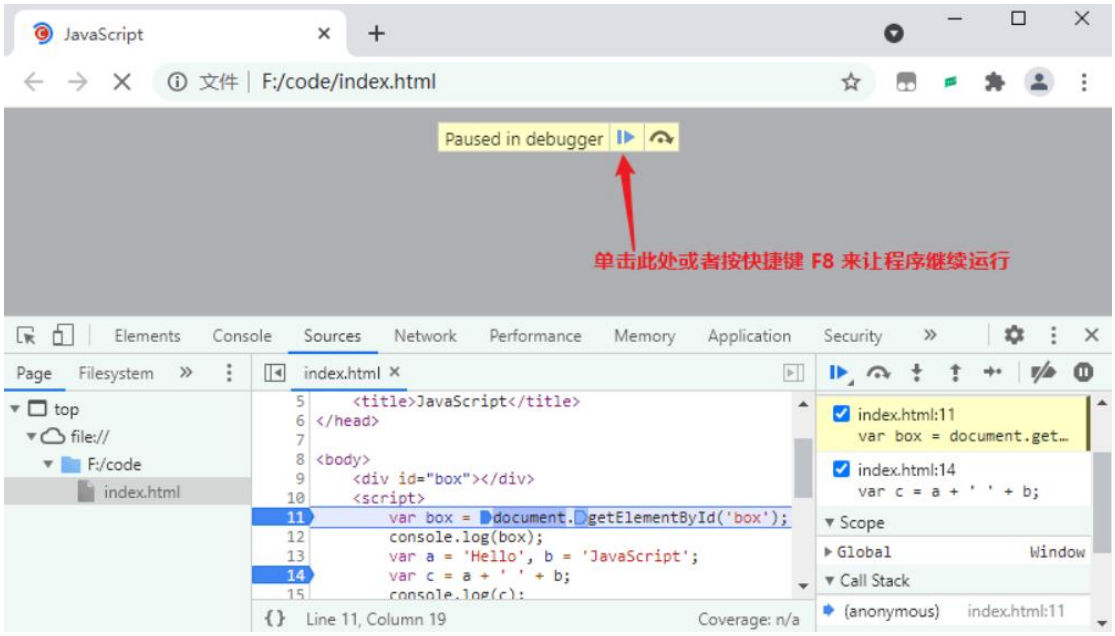
2>打断点

给 js 代码打断点非常简单，只需要单击要调试代码前面的行号即可，若行号被标记为蓝色，则说明已经成功打了断点，如下图所示（在代码的第 11 行和第 14 行打了断点）：



3>断点调试

打好断点后，刷新页面即可进入调试模式，代码执行到断点的位置会暂停，此时我们可以点击页面中的箭头或按 F8 来使代码继续执行到下个断点，如下图所示：

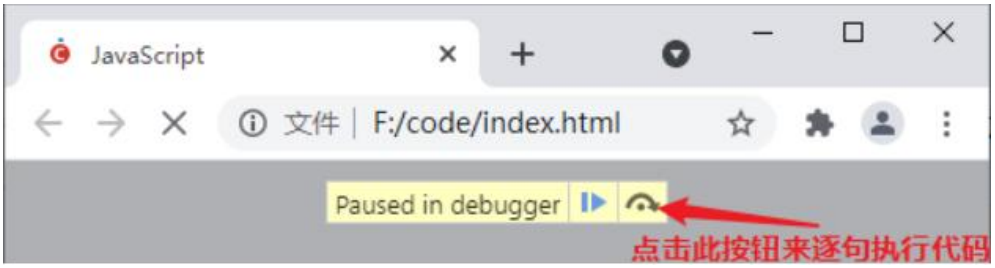


调试过程中，会在调试工具的最右侧的 Scope 栏显示一些数据。此外，还可以在最右侧的 Watch 栏中录入要调试的变量名，这样在调试过程中就能实时看到代码运行中变量的变化。

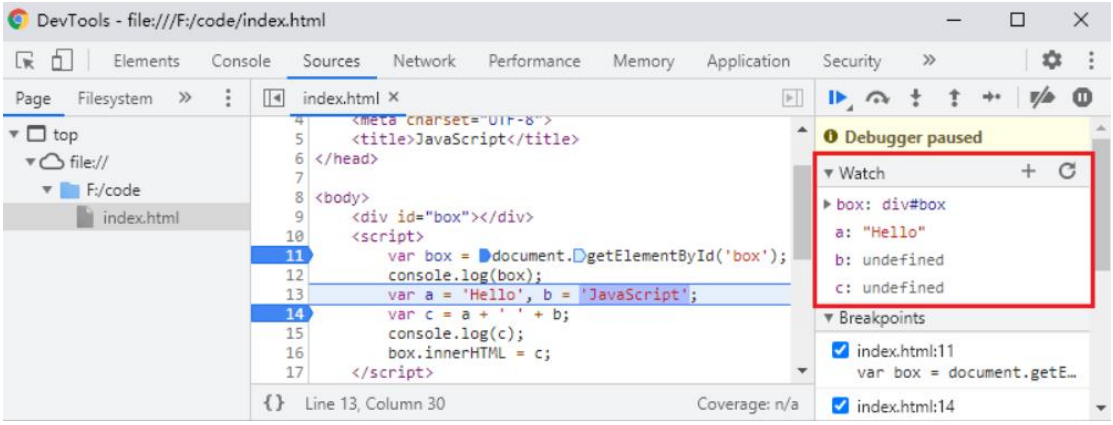
4>逐条语句执行

在调试过程中，我们还可以选择让代码逐句执行，只需要点击下图所示的按

钮，或者按 F10 即可：



逐句执行配合在 Watch 栏中录入要调试的变量，能够更清晰的看到变量在代码运行过程中的变化：

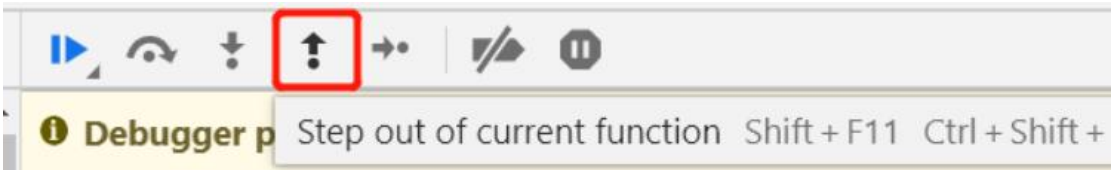


5>单步调试和函数调试

单步调试：又叫普通调试。点击 Step over next function call 就是程序逐步调试，每点击一次，就会按照代码执行顺序，向下执行一句代码。

函数调试：如果亲手尝试过单步调试的小伙伴就会发现，单步调试其实并不能满足我们找 bug 的需求，因为单步调试是不能进入函数体内，我们也就不能跟踪函数体内变量的变化。所以我们接下来就来学习下第三个按钮，step into next function call 按钮

但是在我们已经追踪到想要的变量变化时，函数体内的内容又很多，单步调试到函数结束就很浪费时间。这里就可以使用我们今天学习的第四个按钮，step out of current function call 跳出当前函数体，跳出到之前进入函数体的代码位置。



3.3.3 debugger 关键字

除了可以借助浏览器的调试工具来给代码设置断点外，也可以使用 debugger 关键字在代码中设置断点（类似于 PHP 中的 die 和 exit），效果与在调试工具中设置断点是一样的，示例代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript</title>
</head>

<body>
    <div id="box"></div>
    <script>
        var x = 15 * 5;
        debugger;
        document.getElementById('box').innerHTML = x;
    </script>
</body>
</html>
```

运行上面的代码，浏览器会自动进入调试模式，并在执行到 debugger 关

键字时（代码第 12 行）暂停。