# **[vscode 使用经验，让vue开发跳转定义和webstorm一样](https://juejin.cn/post/6844904098538455047)**

[](https://juejin.cn/user/1873223545523101)

[https://juejin.cn/post/6844904098538455047https://juejin.cn/post/6844904098538455047](https://juejin.cn/post/6844904098538455047https:/juejin.cn/post/6844904098538455047) https://juejin.cn/post/6844904098538455047

[海天酱油\_htSauce IMG_257](https://juejin.cn/user/1873223545523101" \t "https://juejin.cn/post/_blank)

2020年03月21日 10:17 ·  阅读 15667

关注

https://juejin.cn/post/6844904098538455047

自己个人开发的时候都喜欢用webstrom，webstrom各种细节，定义跳转，样式，都非常棒，奈何公司内网开发，逼着我用vscode，在新公司用vscode开发三周之后的今天（周六），我爆发了。

下面是我必须推荐的几个插件

## 一、插件推荐

1、Auto Rename Tag

修改html标签自动补全，改了首标签，尾部自动变。****webstrom自带****

2、Code Runner

自动检测packjson文件执行npm命令。****webstrom自带****

3、样式Material Theme

webstrom也有这样的插件，但是vscode丑好多

4、eslint

必须的

5、GitLens — Git supercharged

可以查看git所有提交记录，并且代码上面自动有提示

6、Prettier

必须的，注意有一个配置你不开启是没法自动格式化的，并且必须自己按保存，然后自动格式化。这个比webstrom好

注意"editor.formatOnSave": true,

该配置是自动保存的关键，因为安装插件是没有自动开启的，也不知道配置哪个，这个自己放在s

7、Settings Sync

把你的配置信息同步到git上面，这样就不会丢失信息了，换电脑都不怕，注意一定要记得保存秘钥和gistid

****提供一个额外的方式，你可以把你的.vscode文件保存下来，然后移植到新电脑，这样也可以转移你的所有配置****

8、vetur

写vue肯定要有

9、Visual Studio IntelliCode

这个还没使用，好像是可以记录你的代码习惯

10、vscode-elm-jump

常规的代码跳转定义，按ctrl，和webstrom一样

11、Vue CSS Peek

和webstrom一样，按ctrl可以跳转css定义，按ctrl，和webstrom一样

12、vue-helper

让你在vue单文件里面的变量函数跳转定义，按ctrl，和webstrom一样

13、Bracket Pair Colorizer 2

标识各种不一样的括号，挺好用的

14、Image preview

绝对路径下直接预览图片，vue文件的@就没用了

15、别名路径跳转，

名字就是这个，直接搜索就行了，提供关于import定义的跳转

## 二、快捷键

常规快捷键我没啥可说的，但是我们在快捷跳转定义之后，是不是想着再跳回去。webstrom上面直接ctrl+z就回去了。

vscode：alt+左键或者右键（上下左右的那个），就可以往前或者往后了。

评价：

vscode虽然轻便，但是和webstrom相比，真的差好多。细节，所有的细节都能让开发者提升开发效率

差不多先这些吧，上面10,11,12，用vscode的必须要装，这样就可以提供基本上和webstrom一样的代码定义跳转功能了。我是弄了一早上，搞这个。

# [2022年了，该学会用VSCode debug了](https://juejin.cn/post/7080135520902184997)

[](https://juejin.cn/user/1626932940390279" \t "_blank)

[Yayoiqz IMG_349](https://juejin.cn/user/1626932940390279" \t "_blank)

2022年03月28日 21:02 ·  阅读 12128

关注

# 前言

VSCode作为目前使用人数最多的IDE，在前端开发者中也是十分受欢迎的。它免费、开源、还具备许多强大的功能，例如智能提示、插件商店、集成Git等等，但除此之外还有一个被许多开发者忽略的功能——运行和调试（Run and Debug）。

你可能会说，****要什么Debug，我**console.log**一把梭走天下****。没错，这也是目前很多人在使用的调试方法。但是既然VS Code具备这样的功能，试一试，或许会有新的收获呢~

# 简介

VS Code的关键特性之一就是它具有****强大的调试（debug）功能****，内置的调试器（****debugger****)可以帮助开发者快速的编辑、编译和调试。

VS Code内置的调试器支持Node.js运行时，并且能够调试JavaScript、TypeScript、以及任意其他****能够编译成**JavaScript**的语言****。

如果想要调试其他的语言和运行时，包括但不限于PHP、Ruby、Go、C#等等，可以在扩展商店中寻找相关的Debugger扩展并且安装它，这里我们就不过多的叙述了。

# 简单调试

为了方便我们理解常用功能，直接进行调试来学习相关功能更容易上手。这里我们用一个简单的Node.js项目作为例子进行调试。

首先准备一个app.js:

let msg = 'Hello world';console.log(msg);let numA = 6;let numB = 13;let num = numA + numB;console.log(num);复制代码

这样我们就有一个简单的Node.js程序了。

接下来我们点击左侧菜单中的运行与调试图标（图标是一个虫子+开始，顾名思义debug&run），这里就是初始化的面板；然后我们在代码中需要打断点的地方加上断点：

直接点击运行和调试：

程序跑起来了，并在断点处也停下来了。调试面板也被激活，显示了变量、监控、调用堆栈、断点的面板。点击上方调试操作栏的第一个图标（继续，快捷键是F5），程序会走到下一个断点，并且变量面板中的msg变量会进行更新：

继续点继续，直到跳出最后一个断点，我们的调试步骤就结束了。这就是一个最简单的Node.js程序的调试过程。

当然我们实际开发中肯定不会有这么简单的程序，所以接下来我们来详细介绍下调试中的相关功能。

# 功能介绍

虽然上面的例子是Node.js项目，但是对于其他的调试器，大部分的概念和特性也都是通用的。

## 运行面板和菜单

在上面的例子里我们已经看到过运行面板了。点击左侧的“运行和调试”图标即可打开面板。运行面板显示了所有关于运行和调试的相关信息。

如果还没有配置launch.json，那么VS Code就会显示初始状态的面板。例子中我们就没有进行配置，所以显示就是初始状态：

除了左侧的图标，也可以使用顶部的顶级菜单 运行（Run)。这里的命令基本和面板中的一致：

想不起快捷键的时候也可以在这里看看~

## 启动配置

上面的例子中我们选择了“运行和调试”，VS Code直接使用了内置的Node.js调试配置启动了调试步骤。 然而大部分场景中，我们不会有这么简单的调试。此时创建一个自定义的launch配置文件就十分有必要了，我们可以在配置文件中保存一些调试的细节。

VS Code把调试配置信息保存在.vscode目录下的launch.json文件中（.vscode目录一般存在于项目的根目录下）。

要创建一个launch.json文件，在运行初始化面板中点击“创建一个launch.json”：

VS Code会去尝试自动检测当前调试环境。如果它失败了，我们就需要自己手动选择：

选择Node.js后，VS Code会自动生成配置文件以及.vscode目录。这里是Node.js默认的launch.json:

{

// 使用 IntelliSense 了解相关属性。

// 悬停以查看现有属性的描述。

// 欲了解更多信息，请访问: https://go.microsoft.com/fwlink/?linkid=830387

"version": "0.2.0",

"configurations": [

{

"type": "pwa-node",

"request": "launch",

"name": "Launch Program",

"skipFiles": [

"<node\_internals>/\*\*"

],

"program": "${workspaceFolder}/app.js"

}

]

}复制代码

这里需要注意的是，对于不同类型的调试器，launch配置中的属性可能都是各不相同的。可以使用VS Code内置的智能提示功能去查看有哪些属性可用，hover属性就可以看到帮助信息。

不要想当然地认为某个调试器中存在的属性放到其他的调试器下中也能起作用。调试前要确保所有配置都是有意义的。

### launch和attach

在VS Code中，有两种核心的调试模式，Launch和Attach，他们为开发者提供两种不同类型的工作流。

最简单的方式来理解这两种工作流：

* launch配置可以理解为VS Code启动调试程序的说明书；
* attach配置则是如何将VS Code的调试器连接到已运行的应用程序或进程的方法；

VS Code的调试支持用调试模式启动一个程序，或者用调试模式附加到一个已经在运行中的程序。使用哪种调试配置，取决于我们的调试需求。

### launch属性

当你在launch.json中配置了type属性的值后，就可以使用VS Code的智能提示（command+I）来查看所有可用的属性：

下面的属性是每个launch配置文件必须具备的：

* type - 调试器的类型。
* request - 请求类型，目前支持launch和attach
* name - 在Debug启动配置下拉菜单中显示的名字

下面是一些可选的配置属性：

* presentation - 在 presentation 对象中，用 order, group, 和 hidden 属性可以在调试配置下拉菜单以及快速选择调试选项中进行排序、分组以及隐藏配置。
* preLaunchTask - 在开始一个调试会话前可以启动一个任务
* postDebugTask - 在结束调试后启动一个任务
* internalConsoleOptions - 这个属性用来控制调试控制台面板的可见性。
* debugServer - ****只为调试扩展的作者使用（for debug extension authors only）**** 这个属性允许你去连接到一个特定的端口而不是去启动调试适配器。
* serverReadyAction - 如果要在调试中的程序向调试控制台或集成终端输出特定消息时在web浏览器中打开URL。

许多调试器支持以下的属性：

* program - 当启动调试器时要运行的可执行程序或文件
* args - 传给程序用来调试的参数
* env - 环境变量 ( null 值可以用来 "undefine" 一个变量)
* envFile - 具有环境变量的.env文件的路径
* cwd - 用来寻找依赖和其他文件的当前工作目录
* port - 连接到正在运行的进程时的端口
* stopOnEntry - 当程序启动时立即停止
* console - 使用哪种控制台，例如 internalConsole, integratedTerminal, or externalTerminal

### 变量替换

VS Code在launch.json中提供了许多有用的变量，并在启动时支持字符串内部的变量替换。例如，${workspaceFolder}给出了工作区文件夹的根路径，${file}给出了在活动编辑器中打开的文件，${env:Name}给出了环境变量Name的值。可以查看[变量引用](https://link.juejin.cn/?target=https://code.visualstudio.com/docs/editor/variables-reference" \o "https://code.visualstudio.com/docs/editor/variables-reference" \t "_blank) 来查看所有的预定义变量。

## 调试操作

当启动调试会话后，调试工具栏会显示在顶部：

从左到右依次是：

* 继续/暂停 F5：执行到下一个断点
* 单步跳过 F10：从断点处执行单步调试
* 单步调试 F11：进入函数内部
* 单步跳出 shift+F11：跳出函数内部
* 重启shift+command+F11
* 结束shift+F5

## 断点(Breakpoints)

点击编辑器左边距可以切换断点（快捷键为F9）。在运行视图中的“断点”部分中，可以对断点进行更多的控制。

编辑器左侧的断点一般展示为红色实心圆，不可用的断点为灰色实心圆。 如果调试器支持在不同的错误or异常中退出，在断点部分中也会可用。

## 记录点(Logpoints)

记录点是断点的一个变种，它不会break到调试器中，但是能够往控制台中打印一些信息。当调试不能暂停or停止的生产服务时，记录点特别有用。

记录点显示为一个菱形的图标。记录信息是解释文本但是也可以使用可计算的表达式（用大括号包裹）

就像普通的断点一样，记录点也可以启用、禁用，并且通过条件来控制。

# Node.js调试

在VS Code中调试Node.js有三种方法：

* 在VS Code的集成终端中使用auto attach来调试程序；
* 使用JavaScript调试终端
* 使用launch配置，或者attach到其他的程序

## Auto Attach

当启用****Auto Attach****功能时，Node调试器会自动附加到VS Code中启动的Node.js进程上。

Command+Shift+P打开输入Auto Attach即可开启功能：

这里的三个选项，我们选择智能(smart)就可以。开启后重启一下VS Code，启动程序后调试器就会自动附加上。此时下方的状态栏会显示auto attach的状态，也可以点击进行更改。

## JavaScript集成终端

和auto attach相似，用JavaScript调试终端可以自动的调试你在终端里运行的任何Node.js程序。在终端的下拉选择框中选择****JavaScript Debug Terminal****即可：

## Launch配置调试

通过配置launch.json进行调试是比较传统的调试方式，根据自己的项目代码需求进行配置，灵活性比较高。

除了可以直接通过node命令启动Node.js程序，我们也可以通过配置使用npm或其他工具进行调试。

* 任何PATH上的可用程序（例如npm、gulp等等）都可以用在runtimeExecutable属性中，并且参数可以放在runtimeArgs中传入。
* 如果npm脚本或其他工具隐式地指定了要启动的程序，则不必设置 program 属性。

我的上一篇文章 [记一次不完全的源码调试之npm install](https://juejin.cn/post/7074934981625118750" \o "https://juejin.cn/post/7074934981625118750" \t "_blank) 就是通过这种方式进行调试的。

# SourceMap调试

VS Code的JavaScript调试器支持可以帮助调试转义语言的source map（例如Typescript、压缩or混淆过的JavaScript等）。使用source map就可以在源码中单步调试或者设置断点。

如果原始代码中不存在source map，或者source map被破坏，无法成功映射原始代码和生成的JavaScript，则断点显示为未验证（灰色空心圆），如下图：

source map功能由sourceMaps属性控制，默认为true。要使用source map进行调试的前提是，我们的项目代码中要能生成source map文件才可以使用，所以如果使用Typescript、babel、webpack等，需要进行相应的配置生成source map文件后再进行对应的配置使用。详细可以查看[VSCode-Source map](https://link.juejin.cn/?target=https://code.visualstudio.com/docs/nodejs/nodejs-debugging%23_source-maps" \o "https://code.visualstudio.com/docs/nodejs/nodejs-debugging#_source-maps" \t "_blank) 相关的官方文档。

如果想要使用VS Code调试vue项目，可以参考[# 如何让 Vue、React 代码的调试变得更爽](https://juejin.cn/post/7071219293249077256" \o "https://juejin.cn/post/7071219293249077256" \t "_blank)。该配置在vue2的项目中可用，vue3目前还没实践成功。

# 高级断点

最后介绍一下VS Code中的一些高级断点功能，日常调试可能用不到但是可以了解一下。

## 条件式断点

VS Code中一个强有力的调试功能是具有****基于表达式、命中次数或两者的组合设置条件****的能力。

* 表达式条件：每当表达式的计算结果为true时，都会命中断点。
* 命中次数：“命中次数”控制断点需要被点击多少次才能中断执行。

你可以在创建断点或修改现有断点时，添加条件or命中次数。在这两种情况下，都会打开带有下拉菜单的内联文本框，可以在其中输入表达式：

当i=3时，会在断点处停下：

或者设置命中次数：

命中次数等于5时在断点处停下：

如果调试器不支持条件断点，那么就不会显示添加和编辑的菜单。

## 内联断点

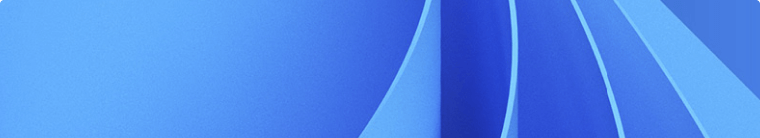
只有当执行到达与内联断点关联的列时，才会命中内联断点。当调试在一行中包含多条语句的小型代码时很有用，例如for循环：

用shift+F9添加内联断点。内联断点在编辑器中内联显示。

# 总结

VSCode的调试功能还是十分强大的，上面介绍的只是一部分内容，官方文档中还有很多内容可以探索。

虽然对于vue.js这种有自己的devtool的框架，直接在浏览器调试更加直观，但调试其他的Node.js程序，用VS Code效率还是很高的。

**选择你感兴趣的技术方向**

后端

前端

Android

iOS

人工智能

开发工具

代码人生

阅读

跳过

上一步

至少选择1个分类

# **《管锥编》序|原文|翻译|赏析**

来源：网络转载    作者：未知    更新于：2019-05-13 23:18:05

[手机版](https://m.pinshiwen.com/gsdq/xuba/2019051343058.html)

瞥观疏记，识小积多。学焉未能，老之已至!遂料简①其较易理董者，锥指管窥②，先成一辑。假③吾岁月，尚欲赓扬④。又于西方典籍，褚⑤小有怀，绠短试汲⑥，颇尝评泊⑦考镜⑧，原以西文属草，亦思写定，聊当外篇。敝帚之享，野芹之献⑨，其资于用也，能如豕苓桔梗乎哉⑩?或庶几比木屑竹头尔⑾。命笔之时，数请益于周君振甫，小叩辄发大鸣，实归不负虚往， 良朋嘉惠，并志简端。

1972年8月

初计此辑尚有论《全唐文》等书5种，而多病意倦，不能急就。已写定各卷中偶道及“参观”语，存而未削，聊为异日之券。

1978年1月又记

( 《管锥编》， 中华书局， 1979年3月第一版。)

**注释** ①料简——料，整理、处理。《广雅·释诂二》： “料，理也”。简，通“柬”，即选择。《书·冏命》： “慎简乃僚”。②锥指管窥——典出《庄子·[秋水](https://www.pinshiwen.com/wenfu/guwen/201902072554.html" \t "https://www.pinshiwen.com/gsdq/xuba/_blank)》篇。《[庄子](https://www.pinshiwen.com/cidian/zhuangzi/" \t "https://www.pinshiwen.com/gsdq/xuba/_blank)》原文为：“子乃规规然而求之以察，索之以辩，是直用管窥天，用锥指地也， 不亦小乎!” ③假——给与。《汉书·龚遂传》：“遂乃开仓廪，假贫民”。颜师古注：“假，谓给与。” ④赓扬，继续。《尔雅·释诂》 “赓、扬，续也。” ⑤褚——音zhu，囊，袋。《集韵·语韵》：“褚，囊也。”《左传·成公三年》：“荀罃之在楚也，郑贾人将寘诸褚中以出。” ⑥绠——汲水桶上的绳索。《左传·襄公九年》：“具绠缶。”杜预注：“绠，汲索;缶，汲器。”[韩愈](https://www.pinshiwen.com/shiji/hanyu/" \t "https://www.pinshiwen.com/gsdq/xuba/_blank)《秋怀》诗：“汲古得修绠。”《荀子·荣辱》：“短绠不可以汲深井之泉，知不几者不可与及圣人之言。”钱先生此处“褚小有怀，绠短试汲”用的《庄子·至乐》篇典。《庄子》原文为“褚小者不可以怀大，绠短者不可以汲深。” ⑦评泊——即评说， 评论。宋朱耆寿《端鹤仙·寿秦伯和侍郎》词：“教公议，细评泊。自和我以来，谋国多少，萧曹卫霍”。明叶盛《水东日记·戏言取怨》：“戏言辩口，轻为评泊，亦足以取怨，得罪于人。” ⑧考镜——参证借鉴。明唐顺之《吏部郎中林东城墓志铭》： “日以朱墨点记其向意，臧否醇杂，以自考镜。”清[龚自珍](https://www.pinshiwen.com/shiji/gongzizhen/" \t "https://www.pinshiwen.com/gsdq/xuba/_blank)《祀典杂议》之四：“方今休隆时，正宜差等百王，考镜群籍， 召万灵之祐，锡九流之福。” ⑨芹献——犹芹意。《列子·杨朱》：昔人有美戎菽， 甘枲茎，芹萍子者，对乡豪称之。乡豪取而尝之，蜇于口，惨于胶。众晒而怒之，其人大惭。”旧用“芹献”或“献芹”为自谦所献菲薄，不足当意之辞。⑩豕苓——即猪苓。药草，可治渴。《庄子·徐无鬼》：“药也，其实堇也，桔梗也，鸡癕也， 豕零也。”陆德明释文：“豕零，……一名猪苓，根似猪卵，可以治渴。” ⑾木屑竹头——典出《晋书·陶侃传》： “侃性聪敏，勤于吏职。……百姓勤于农殖，家给人足。时造船，木屑及竹头悉令举掌之，咸不解所以。后正会积雪始晴，听[厅]事前余雪犹湿， 于是以屑布地。及桓温伐蜀， 又以侃所貯竹头作丁[钉]装船。其综理微密， 皆此类也。”

**赏析** 西方一演讲名家言：10分钟的演讲，需准备两个月;半个钟头的演讲，要准备两星期;两个小时的演讲， 只要准备两个晚上就可以了。演讲的时间愈短，愈要精彩。因为演讲者要在有限的时间内用简练的语言和别具一格的形式表达出自己的意见，展现出自己的风采。演讲的这一规律似乎可以用到写序、跋一类的文章上来。有时文章愈短，就愈要精致而有蕴味，也就愈能见出作者的功力。钱钟书先生的序文多为此种类型。《围城》、《人·兽·鬼》、《写在人生边上》、《谈艺录》、《七缀集》等的序言无不写得短小精道，蕴味深永。《管锥编》可谓煌煌巨著，而其序言却只有一百多字，是这类短小精致的序的典型代表。

这篇小序篇幅虽短，内容却颇丰。首先，提示了本书的特点并解释了《管锥编》的书名。“瞥观疏记，识小积多。”说明本书是作者长期以来阅读钻研典籍的发现、体会的积累。它不属于有系统的理论建构，而是缘《容斋随笔》、《日知录》一类的考证、札记或随笔。体现了作者打破体系建构，摒弃一切玄谈空议，而认为“诗、词、随笔里，小说、戏曲里，乃至谣谚和训诂里，往往无意中三言两语，说出了精辟的见解，益人神智;把它们演绎出来，对文艺理论很有贡献”(《七缀集》33页)的一贯主张。就《管锥编》这一书名，钱先生用《庄子·秋水》篇中“用管窥天，用锥指地”的典故，说自己的发现和见解是“锥指管窥”，解释了书名的涵义，并进一步以谦虚的方式说明该书的性质和特点。另外，表明了此书写作时的情况和作者的心情。“学焉未能，老之已至!”“假吾岁月，尚欲赓扬。”作者深感无谓的政治运动已经耗费了太多的宝贵时间，表现出要与时间赛跑，要把自己的学识积累传与后人的急切心情和“烈士暮年，壮心不已”的情怀和志向。第三，表现了作者一贯的谦虚的态度和对曾帮助过自己的朋友的谢意。钱先生的学问博大精深，但他却自谦是“褚小有怀，绠短试汲”，把自己的著作说成是“锥指管窥”， 比喻为“敝帚之享，野芹之献，其资于用也，能如豕苓桔梗乎哉?或庶几比木屑竹头尔”。这是钱先生一贯的风格。他幼承家学，其渊博睿智及惊人的记忆力是大家公认的。他的《谈艺录》是众口皆碑的杰作，而他却说：“自叹颛愚，深惭家学，……未耄善忘，不醉多谬;蓄疑莫解，考异罕由。” (《谈艺录·序》)并把他的《旧文四篇》和《也是集》称为“半吊子”、“二毛子”(《七缀集·序》)。当然，在这篇小序中，把《管锥编》的作用比作能给人解渴治病的豕苓桔梗和确有实用的木屑竹头，在谦虚当中也隐隐透露出作者对此书的自信。

这篇小序用简洁雅致的文言写成，并于文中穿插运用典故，显出精练古雅，言简意赅，蕴涵丰富，意味隽永的特色。