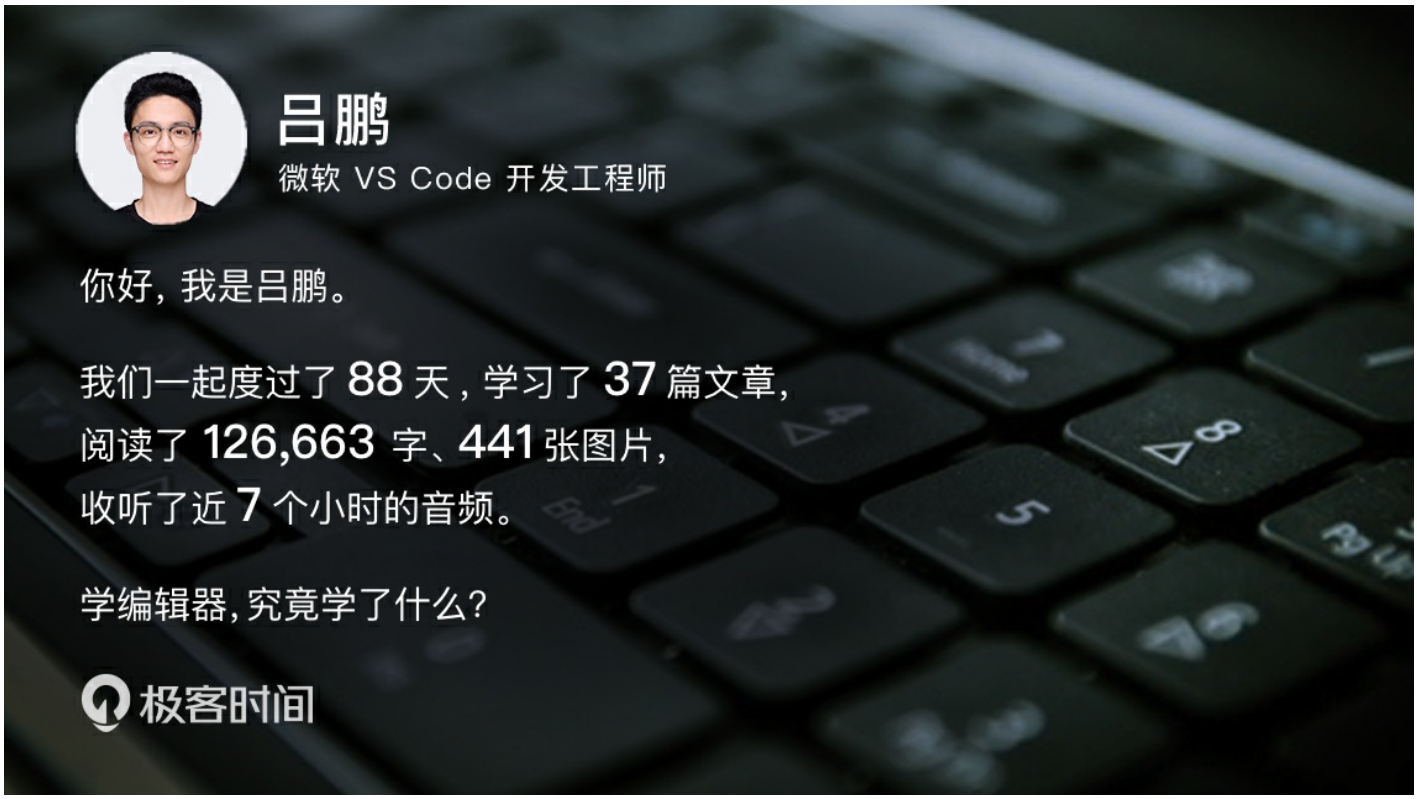
结束语讲学编辑器，究竟学了什么



连我自己也不敢相信，三个月的专栏写作，到这里就结束了。专栏的第一篇文章《学编辑器，到底应该学什么？》仿佛就像是昨天才写过一样，每一行文字、每一次修改都还历历在目。

很多作者在写完专栏后都表示写专栏其实是一个非常痛苦的过程，写文字比写代码难太多了。如果不是亲身经历了这一切，我也很难相信这一点。不过在极客时间编辑们的支持和帮助下，我坚持了下去，完成了这个挑战。

你可能不禁要问了，写专栏究竟难在哪里呢？难道这个专栏的主题不是作者你非常熟悉的内容吗？

其实我觉得大部分的痛苦，就是来自于“我觉得我足够了解”。我不仅是编辑器的深度用户，还在维护一个越来越流行的编辑器，身边的有些同事在这个领域已经耕耘了快二十年了，没道理我不了解这个主题啊。

但事实并非如此，开发工具从软件开发的第一天就存在了，几十年积累下的知识太多了。所以，在写每一篇文章的时候，我都会思考：这个功能的作用是什么？为什么要设计成这样？使用它的最佳实践是什么？越思考我越后背发麻，因为一些答案我竟不那么明了，甚至还有好一些地方，VS Code 设计的也不算理想。

但是每次看大家的留言时，我都会特别激动，因为好多读者都有很不错的见解，总结能力也很强太，让我很意外又很惊喜。“授人鱼不如授人以渔”，既然如此，我不妨把我写专栏过程中的一些思考分享给你，顺带再给专栏做一次梳理。这样，即使专栏结束了，你也可以继续编辑器的学习、思考和总结。

那么，学编辑器，我们究竟学了什么？从专栏目录来看，我基本上是根据 VS Code 的功能和组件来一一介绍的。但其实，这其中又包含了两个隐藏的主题：效率和定制。

效率

专栏一开始就不断强调效率这个概念，因为开发工具的目的，就是帮助用户更好地完成工作，效率肯定是第一位的。那么，怎样使用 VS Code 才能最高效呢？

这个问题的答案，一定是因人而异的。因为影响因素非常多，包括自己的计算机使用习惯、工程项目的类型、团队合作的方

式，等等。所以在介绍每个功能时，我都会试着去介绍这个功能的出发点，要解决什么问题，又有哪些局限性。比如说：

- **快捷键**。VS Code 对自己的定位是 keyboard oriented，意思是你可以只通过键盘就完成大部分操作。如果是操作编辑器中的内容，能够熟练使用这些快捷键的话，一定是可以优化代码书写效率的，Vim 快捷键就是一个很好的例子。不过这并不一定意味着，使用快捷键就能够最高效地完成所有操作，因为这不是它的目的。所以像代码调试、版本管理等章节，我几乎没有怎么介绍快捷键，虽然你依然可以在命令面板中找到它们，但是显然相较而言，通过鼠标能够更轻松地使用这几个组件。
- **代码跳转、智能补全和代码折叠**。这些功能，都依托于一个完善的语言服务。如果语言服务不够智能，那这些功能几乎就没法正常工作，更不要提效率了。由此，你不妨研究研究，对于你的项目、语言和框架，VS Code 有很好的插件支持吗？如果答案是 Yes，那么这些部分就值得多花时间研究；如果答案是 No，那么你就得看看，有什么“土办法”能达成一样的效率。
- **代码片段（Code Snippet）和搜索**。如果你擅长书写代码片段和使用搜索的话，即便没有智能语言服务，效率也一样杠杠的。
- **版本管理、集成终端和调试**。这些组件之所以存在于工作台之上，是因为 VS Code 希望给你提供一个更好的交互界面，本质上，它们的底层大部分工作，都可以通过命令行在系统的终端里实现。

定制

除了效率以外，可定制化也是一个非常重要的主题。因为，要想让自己更高效地工作，要么改变自己，去适应编辑器的各个功能，要么就定制和修改编辑器，让它更符合自己的工作习惯。VS Code 并不像 Eclipse 和 Atom 那样，允许你修改工具的每个功能，但是它依然有足够多的定制项：

- 你可以修改快捷键配置，如果 VS Code 的默认快捷键选择跟你的肌肉记忆不相符，那就将它改掉好了。
- 你可以定制主题，修改各个部件的颜色定义。
- 你可以定制各个组件的位置、可见性。
- 如果 VS Code 自带的功能已经不能够满足你的使用，那么你也可以自己定义语言，通过插件来调整 VS Code 的使用方式。

相信了解了这两条暗线，当你重温这个专栏的时候，或者在专栏中找寻某个知识点的时候，就方便多了。如果你还不满足于专栏的内容，那么最佳的学习方式一定是：**追着 VS Code 的更新和开发进程，实时地了解 VS Code 的动向**。

VS Code 的所有开发计划，都放在了 GitHub 上。每个月，VS Code 都会出一个Iteration Plan，详细记录这个月要做的每个功能。而每个月发布新版本后，VS Code 也都会出一份 Release Note，这份报告会以图文的形式，介绍新版本的每一个功能和大的变动。

如果你只是希望更好地使用 VS Code，那么上面的这些内容，就能够保证你获得 VS Code 的第一手资料了。最好的内容，一定是第一手的、通过自己的阅读得来的，再详细的专栏，也没法像 VS Code 的 Release Note 和开发计划一样细致地介绍每个功能了。

不过也很有可能，你并不满足于此。身为开发者，可能都希望 “知其然，又知其所以然”，要是能够深入 VS Code 的代码看看，那就再好不过了。我在专栏里介绍每个组件时，都会顺带提一下它们的英文名字，比如Workbench、Code Snippet、Multi Root Workspace、Integrated Terminal，这些名字相信你已经不陌生了。而知道了这些名字，要想到 VS Code 的项目里，找到它们对应的代码可就太容易了。VS Code 的代码也托管在 GitHub 上，根目录下有各种项目的配置，而其中有两个文件夹至关重要：

- src，就是 VS Code 的核心代码；

- extensions，就是 VS Code 默认自带的插件实现。

extensions 无须多说，它们跟插件市场上的代码没什么区别，阅读完专栏的插件开发部分再来看它们可就毫无难度了。如果你想知道一款成熟的插件应该长什么样，想通过学习插件代码来了解 API 的使用，那么 extensions 文件夹就是很好的开端。

src/vs 这个子文件夹，则包含了 VS Code 全部组件和功能。其中：

- base、code 和 platform，是一些基础部件和服务，比如 list view，inputbox 等，这些都是完成 VS Code UI 和组件的基础零部件。
- editor，就是咱们书写代码的编辑器了。
- workbench/parts，这个文件夹内，存放了 VS Code 工作台上，除了编辑器以外的其他所有组件的代码。而这里面的每个文件夹，都对应了 VS Code 的某个组件。debug 就是调试器，extensions 就是插件管理器，output 就是输出面板，search 就是跨文件搜索等等。

我想，**学习源码，可能是最高级、最硬核的编辑器学习方法了吧**！

最后，感谢所有订阅了这个专栏的读者，以及极客时间的所有工作人员，感谢你们的陪伴和鼓励，我们后会有期。

精选留言

兵戈
感谢老师，收益良多！
2018-12-10 17:12

Harry
期待与您再会！
2018-12-10 07:41

思考问题的熊
谢谢老师，收获非常大
2018-12-10 00:19

CHENG
感谢作者提供的优质内容，前些天还在公司的confluence里面写了一篇结合项目的安利软文。不过在爱尔兰的朋友表示刷极客

时间看到轻微怀疑人生

2018-12-08 20:34