从 VS Code 的历史中可以学到的经验 - 阿里云开发者社区

> VS Code 作为目前使用人数绝对 Top1 的 IDE/Editor（Stackoverflow 2021 调研 ： https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-integrated-development-environment 有 71% 的开发者使用），一定是做对了一些关键的事情才达到今天的规模，如果想做好一个技术性的产品或工具，细细研究，一定能有所收获。

用 VS Code 之父 Erich Gamma 的话讲：VS Code 的成功是“An Overnight Success”，一夜成名。当然这是作者在调侃。

作为一个工具类产品，细节是魔鬼，不存在一招鲜吃遍天。VS Code 也不例外，打磨了 10 年，从 2011 年前身的 Monaco，2013 年后团队差点被解散，2015 年改名 VS Code 后发布，此后一直是稳步增长。

我个人是最初使用了 6 年 Vim（Ruby + JS），然后是切换到 Sublime，短暂的 Atom，到最后使用 VS Code 一直到现在。

大环境上，编辑器是很热门也是传统的赛道，VS Code 本有很多机会泯然众人，但各种原因让它最后一骑绝尘。

接下来一起扒一扒 VS Code 的发展历史。

先了解下作者

VS Code 之父 Erich Gamma 作为设计模式“四人帮”作者之一，软件模式发展的先驱。开发了 Java 圈的单元测试框架 JUnit，更辉煌的是，在 IBM 主导开发了 Eclipse IDE 编辑器，当年最流行的 编辑器 / IDE。可是，Erich Gamma 觉得有点不太对，未来属于 Web，他想在 Web 上打造像桌面端一样的开发体验。

在那个 IE 6 浏览器很香、jQuery 是最先进前端框架的年代，大佬果然是大佬，比一般人往后看 10 年。

同时 IBM 也江河日下，这个时候微软抛出橄榄枝，“入伙吧，未来是云时代，来做 Azure 的 Web 端编辑器”。Azure 刚于 2010 年发布，需要一个 web 端的编辑器。

双方一拍即合！（过程当然并不像预料的那样一帆风顺）

2011 故事开始 – Monaco Editor

因为有 Eclipse IDE 的开发经验，很快撸了一个 Web 端的编辑器。

猜测是因为 Erich 喜欢去摩纳哥旅游，起名叫 "Monaco"。这是一个 纯 Web 的 Editor，VS Code 前身。也是目前 VS Code 使用的 Editor。很快提供给 Azure 用户使用。

Monaco 的特点就是“快“。性能吊打 Ace 和 CodeMirror。我们团队孵化的“SQL 编辑器”也是基于 Monaco。

这也是 VS Code 至今的原则：不使用任何 UI Framework，这是为了追求极致的性能，尽可能接近 DOM，做到每一个性能损耗点都能完全控制。

2013 年：全面转向 TypeScript

JS 语言设计的时候太过仓促，留下很多怪癖，且动态类型导致很多问题在运行时才能发现，开发像 VS Code 这样的项目难堪重任。想一下如果没有 TS 的类型校验，重构一个大型的 JS 项目将会是怎样的灾难！（这一点做 Quick BI 开发的时候深有体会，如果没有 TS 的帮助，我们不可能发展这么快。）

这一次，幸运又一次降临在 Erich。他是同事兼好基友 Anders Hejlsberg 开发了 TypeScript。从 2011 年 Monaco 就使用 TS，感觉越用越好，2013 年决定全面切到 TypeScript。

更巧合的是：据说 Anders Hejlsberg 有一个怪癖，喜欢在一个文件内写所有代码，所以大文件特别多，他也是 Monaco 的重度用户，有他的加持，Monaco 性能更进一步。

因为转到了 TypeScript，给 VS Code 的快速发展打下来坚守的技术基础。

2013 年：差点挂掉的一年

开发了 3 年的工具，虽然性能逆天，但致命的缺点是月活只有 3000 用户！

“你到底为公司创造了什么价值？”Erich 也面临灵魂拷问。

算了一下，用户数至少 x10 倍才能养活团队。但做 Web Editor 大概也只能这些用户，继续搞 Monaco 还是换一个新赛道？怎么办？

2014 年：机遇来了

2014 年，微软开始全面转型到生产力工具和云优先。从只考虑 Windows 平台到要考虑 MacOS、Linux、Windows 跨平台，并全面拥抱开源。

但微软缺少一个跨平台的开发工具，尤其是这个时候 Web 开发者开始迅速增长。

Erich 意识到机会来了。

但 2014 年的浏览器还是不够给力，前端项目越来越重，需要大量的文件处理。不是一个 Monaco 在线编辑器可以搞定的。

Erich 面临艰难选择：继续发展挚爱的 Web 编辑器 Monaco 还是改到 Desktop 版的 Editor/IDE？？？

最终：结果大家都知道了，Erich 选择了开发者更能接受的 Desktop Editor/IDE。

方法也简单，用 Electron 包壳 Monaco。

并改了个网红潜质的名字“Visual Studio Code“，蹭了 Visual Studio 的热度，根正苗红，几个月就搞好了。

2015 年：终于它来了 – VS Code

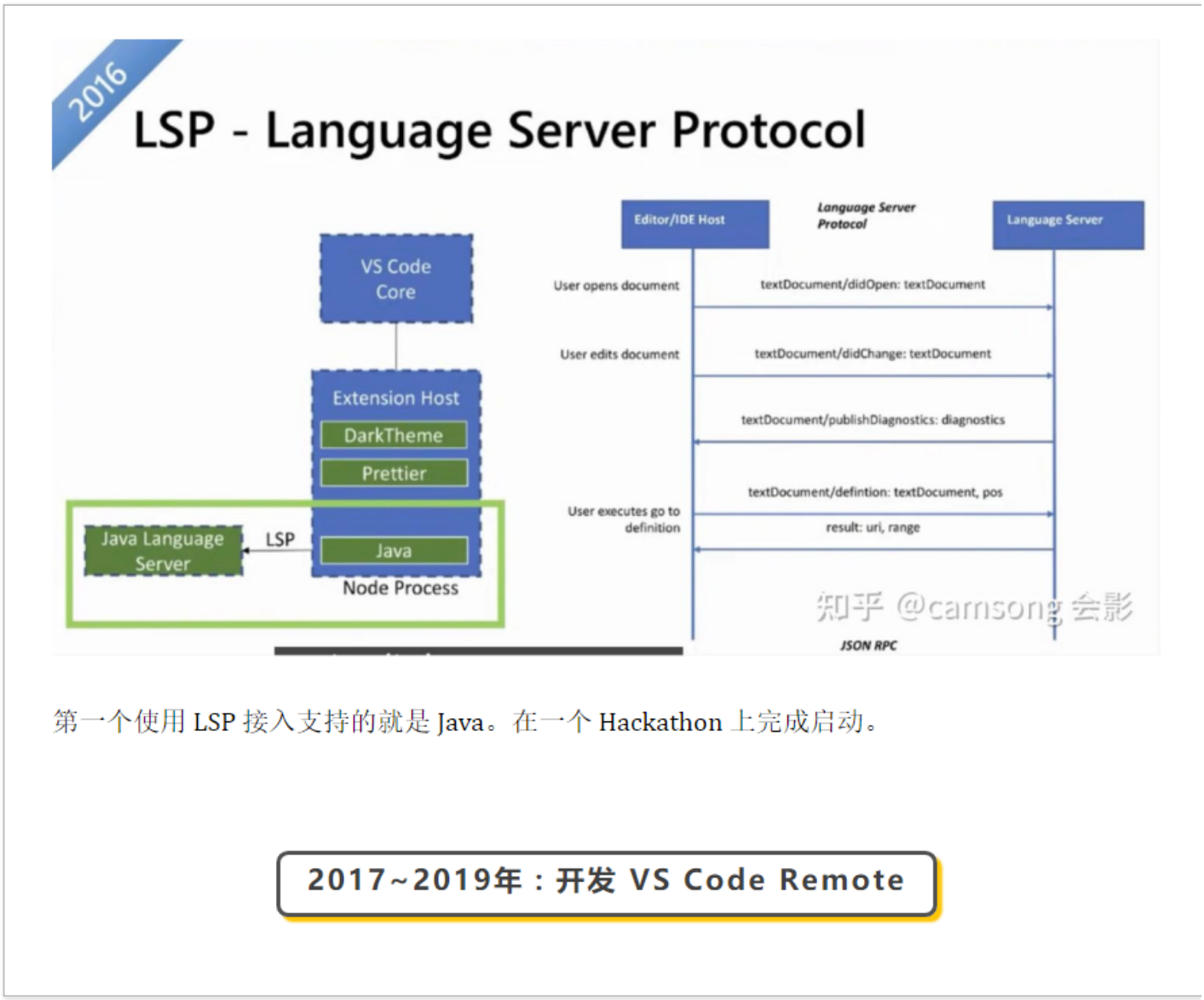
为了给尽可能多的 MS 人使用，提升业务价值，VS Code 选择在 BUILD 大会上正式发布。

当时最成功的一个 demo 是演示 Linux 下调试 .NET 应用。

但当时很多功能停留在 Demo 上，比如很多人期待的 Extension 扩展 API，是在 6 个月后上线的。

2016 年：多语言支持 LSP(Language Server Protocol)

开发过 VS Code 插件的朋友应该很熟悉 LSP，可以扩展其他语言支持。



第一个使用 LSP 接入支持的就是 Java。在一个 Hackathon 上完成启动。

2017~2019 年：开发 VS Code Remote

2017 年开始做 open in VS Code，到后来就是 VS Code Remote。

这样本地只需要一个低配置的浏览器，调试器和命令行都执行的远端服务器上，真正实现多人完全相同的环境下云开发。

因为涉及到跨平台，微软的 WSL(Windows Subsystem for Linux) 帮了大忙，解决了不同操作系统文件系统的差异，不然单是简单的文件路径匹配就是很麻烦的事情。在这之前，Windows 最被开发者吐槽的是差劲的 Shell 支持，虽然有 powershell 但距离 Linux 的 shell 还是差距很大。WSL 通过在 Windows 中运行 Linux 子系统，可以接近 Linux 原生性能执行各种 bash、grep、ls 操作。

2020 年：重返 Web

VS Code Desktop 版本深度依赖了各种 Node.js File API。所以即使是 JS 开发的 Electron 应用移植到 Web 也是需要花不少时间。

2020 年终于发布纯 Web 版的 VS Code。

也是在 2020 年，出现很多在线的 IDE/Editor 开发工具，低代码工具满天飞，很多是基于 VS Code 代码改造。

总结

VS Code 之所以成功，有很多因素。除了 LSP 这个大杀器，我们无法直接照搬，很多方面我们在做产品的时候可以借鉴。

- 性能够 快！
- 对于工具产品来说，任何时候“快”都是竞争力。
- 快 需要有足够的技术积累、技术克制、细节优化。性能就像打地鼠，稍不留神问题就会冒出来，快慢间来回反覆。
- 我们的产品之所以叫 Quick BI 就是看着“快”的竞争力。

- 预见未来，适时、而变
- 要有预见未来的远见，乐观的理想主义，也要能从现实中找到一个可行路径。
- 虽然 Erich 在 10 年前就押宝 Web 编辑器，但当时时机确实不成熟啊，如果不转成 Electron 套壳，活不过 2014 年。
- 预判云计算崛起，Web 崛起，跨平台开发。定好了大方向 Web Editor/IDE，解决了生存问题（帮助微软做跨平台开发），接下来就看谁能扛的久。逐步建立生态，丰富插件。生态和产品是 0 和 1 的问题，如果产品不好也没法做大生态。所以本质还是 VS Code 本身功能够好，架构优雅，方便扩展。
- 大数据时代，海量数据开发和分析面临挑战，Dataphin 和 Quick BI 也是顺时而生。
- 关注用户声音，持续优化，坚持 10 年，做时间的朋友。
- 最让我感觉震撼的是：去年 2021 年 1 月份数据，VS Code 解决了 100K 个 issue。这是时间的味道！此外 NPS 66 分
- 想象一下，修 10 万个 bug 的软件细节能不好吗？也想一想，你的产品解决过多少 issue/bug?
- 工具类产品由于细节过多，这些大部分都不是低级问题。这是距世界产品的差距，还需要继续加油。
- 运气好
- 机会是留给有准备的人，当微软 2014 年宣布转型的时候，恰好需要跨平台编辑器，Monaco 刚好准备好。
- 有 Anders Hejlsberg 这样的超级用户，并带来了 TypeScript。
- 顺应了微软大的转型。在商业公司里做开源，需要解决“业务价值”问题，否则很容易在资源抢夺过程中被拆解。看看 React 团队的人员更换速度。

我们总是容易高估短期，低估长期。

Be patient, be persistent, be fit, be willing to pivot, be lucky.

本文大部分内容来自 Erich 的演讲，推荐观看：VS Code Day Keynote with Erich Gamma（https://www.youtube.com/watch?v=hilznKQij7A ）

有些内容可能不准确，欢迎拍砖。

全文完