



Stanney Yen

[Follow](#)

从阅读国外文章翻译成中文并加上自己心得开始，努力朝着能高效写出有技术价值文章的小小工程师

Jul 5 · 3 min read



Image From: 10 Things I Regret About Node.js — Ryan Dahl — JSConf EU 2018 Video

深入理解Node.js 的设计错误 — 从Ryan Dahl 的演讲中反思

Node.js 从2009 年问世，是Javascript 在Server-Side 应用的 Framework，在去年2017 到2018 年的使用率仍不断在成长。

使用率参考资料：https://foundation.nodejs.org/wp-content/uploads/sites/50/2017/11/Nodejs_2017_User_Survey_Exec_Sum.pdf

此篇文章想整理Node.js 之父Ryan Dahl 在JSConf EU 2018 的演讲内容，并针对Ryan Dahl 说的每一个设计缺失，深入理解后整理出自己的心得，若有任何不对的地方，也欢迎在下方留言：）你的纠正会是我很大的进步动力！

在开始分析Node.js 的设计缺失之前，我们需要来前情提要一下Ryan Dahl 开发Node.js 的动机跟初衷，以及他想要解决的问题，进而会更理解为何在Node.js 开发之初及几个关键时刻时，没有选择去解决这些设计缺失的原因，导致在Node.js 生态圈建立之后，难以弥补这些缺失。

Node.js 开发动机与背景

2009 年时，Intel CPU 的发展已经如火如荼，CPU 使用上效能已相对过剩，Web 应用开发的瓶颈开始出现在I/O 跟网速，而这时Chrome V8 也刚好问世，Google 也同时将V8 Open Source 了出来。

前面有提到Node.js 本身是使用Javascript 语言，而Javascript 的使用在浏览器端蔚为主流，因此延续着Javascript 本身在浏览器上使用的event-driven 的特性，以及Chrome V8 的加持，不难想像Node.js 会朝向event-driven 的特性发展，透过no-blocking I/O 的设计解决I/O 上的瓶颈，也因使用Javascript 语言开发者本来就很多，让Node.js 在发布后迅速窜红。

Ryan Dahl 大约在2012 年时离开了Node.js 的维护，当时他认为Node.js 已经是有达到他想开发Node.js 的目标— 一个易上手把玩的Server-Side JS Framework，且在windows、Mac上都可以轻易地安装执行(透过IOCP)，以及当时也已经有了NPM 的生态系。

不过，在演讲上，Ryan Dahl 说一切都是当时的他想得太天真了XD，仍是有相当多可以持续让Node.js 更完善的工作可以做才是。

Bugs are never so obvious as when you're the one responsible for them

当Ryan Dahl 不再是Node.js 维护者之后，更能察觉到Node.js 存在的问题

哪些问题呢？让我们接着看下去

进入正题，Ryan Dahl 认为Node.js 的设计缺点

#1 放弃原生支援 Promise

Ryan Dahl 在2009 年6 月时在Node.js 中加入Promise，但在2010 年2 月时又将Promise 移除，导致有段时间，若你要使用Promise 需要额外使用#bluebird 版本以外，虽然过不久后又重新加回来，但却影响了许多核心API 的async API 操作。

举例来说fs系列的API仍是使用callback的设计方式而非Promise，当然可以使用类似mz的扩展包来使用Promise版本的核心API，但当时Ryan

Dahl移除Promise的举动，确实造成了核心API的老化问题，得都一一更新成Promise版本才能解决。

#2 没有谨慎思考安全性问题

前面提到Node.js 是以Chrome V8 引擎来执行JS，而V8 是一个非常安全的sandbox，如你使用Google Chrome 一样，你不能轻易的去访问系统资源，但同样使用V8 的Node.js 却能够不需要『授权』，即可访问网路、档案系统，甚至是记忆体资讯。

这确实牵扯到所谓的安全性问题，但这也是在开发便利性上的取舍，只是在资安考量越趋重要的时代，会是Node.js 被质疑的问题之一。其实，很多语言也有同样的问题，但若你是使用Unix-Like的系统，系统层会档掉而已，所以你会不会被询问是否授权的经验，但在目前Windows 上确实是会畅通无阻。

#3 Build System

这项是Ryan Dahl 认为Node 设计上最大的缺失，就是Node 是使用GYP 做为产生建构项目的系统。GYP 是Google Chromium 团队用来建构项目文件让GCC、VSBuild 等编译平台来编译C++ Library 的工具。

在Node 开发早期，Chrome V8 是以GYP 建构系统，而Node 也就沿用了GYP，但不久后Chrome 放弃GYP 转而使用GN，而Node 已经无法挽回。因此Node 成了目前在V8 上唯一使用GYP 的用户，而GN 速度比GYP 快了将近20 倍、文件可读性高且支援许多依赖。

另外Ryan Dahl 也认为应该提供更为简便的接口(FFI)，让开发者想绑定其他系统library 时可以更为简便，而非需要自己写C++ 才能引用这些动态库，早期有许多人建议Ryan Dahl这么做，但都被Ryan Dahl 漠视了。

不过，强大的Javascript生态圈，当然也有人提供了FFI的扩展包:目

#4 Package.json 与Npm 的集权问题

在Node.js 中，可透过require() 方式来引入package，加上NPM 的生态，看似是一个很不错的设计。但其实产生了几问题



Image From: 10 Things I Regret About Node.js — Ryan Dahl — JSConf EU 2018 Video

1. `package.json` 变成了必需品，这是其他语言开发者所没有过的。
2. `package.json` 记录了太多不必要资讯，包含描述、许可证、仓库等。
3. `require` 自动省略副档名方式，导致需要额外的判断这是js? 还是这是ts? 多了需多不必要的判断，增加编译时的负担。
4. Npm 成了集权且标准的中央函式库，若Npm 发生问题将可能引发全球灾难，像是2016 年的left-pad 问题。
5. 当你引入一个package A，你需要在程式码里require A，且 `package.json` 里要宣告此package A 的版本等等资讯，而NPM Database 要有一份A 的程式码，而你的node_modules 也有一份A 的程式码。如果你也是前端开发者，在Web Browser 的使用情境里，JS 要从哪边include 只需要引入URL 或是本地库就好，而不需要上面这些多余的动作。

在写这篇文章的同时，也刚好遇到专案中的某个npm module 相依了一个Github url，但因为Github 的连结暂时坏了无法连上，因为NPM 的极权，也没有其他位置能够取得此library的copy，因此我只能暂时暂停手边工作。很无奈，但这或许刚好也是Ryan Dahl 特别提出这问题的原因。

#5 node_modules

node_modules 里的每一个folder 并没有标准，因此可以放置多余的版本或是任何其他档案跟资讯，这导致增加了模组解析复杂度。

#6 index.js

若有了package.json，其实就不需要默认加载index.js，这确实让模组加载更加地复杂化了。

结论

其实在仔细研究Ryan Dahl 提出的这几个问题后，我认为他所提的问题，都不是非常严重的问题，甚至可以说大部分是优化及管理上面的问题，而Node.js 生态圈也很努力的在解决这些问题，包含mz 库、FFI 这些扩展都能解决上面所提的几个问题。

透过此文章一开始的历史背景知道，Node.js 一开始的开发动机是为了开发出高性能HTTP Server，因此便可以理解上述问题为何在开发初期并没有严谨看待。

透过Ryan Dahl 的演讲与反思，我们可以知道，当你想开发一个framework 或是看待一个framework 是否值得学习或使用时，你可以注意哪些地方也更能欣赏一个framework 的好与坏吧：)

Ryan Dahl 针对上述问题，提出了解决方案— Deno，Deno 结合 Rust+TypeScript+V8，目前都还在萌芽阶段，有机会可以研究一下 Deno 的程式架构，我想经过Node.js 的开发经历，Deno 开发上会更加简洁优美，值得参考。

[ry/deno](#)

deno - A secure TypeScript runtime on V8

[github.com](#)



附上这次演讲连结，有兴趣可以直接至Youtube 观看

小趣事

有件趣事是，Ryan Dahl 说他上次使用Node.js 已经是六个月前的事情了，现在他都使用Go，因为Go 是一个相对Node.js 来说执行速度上较快的Server (毕竟Go 是Compiled language)。

以我目前接案的经验，以及前阵子对Go 的研究上，除非是非常大型的Web 应用，否则在雏形系统或是一般需求的Web 执行速度上Node.js 已经相当够用(当然前提是你足够了解JS)

Ryan Dahl 也在后头补充，在你需要使用dynamic language 的场景时(一些测试计算程式，或是快速想获得测试结果的场景)，Javascript 仍是他认为最好的开发语言。

以上是针对Ryan Dahl 的演讲所整理的心得，喜欢的话可以给我一些掌声!

任何问题欢迎在下方留言：)

