



( / )

[首页 \(/\)](#)[新手入门 \(/getstart\)](/getstart)[API \(/api\)](/api)[关于 \(/about\)](/about)[注册 \(/signup\)](/signup)[登录 \(/signin\)](/signin)

## 精华 来自朴灵大大的 -- Node.js 简史

• 发布于 2 年前 • 作者 Pana (/user/Pana) • 5229 次浏览 • 最后一次编辑是 3 个月前 • 来自 分享

去年12月，多位重量级Node.js开发者不满Joyent对Node.js的管理，自立门户创建了io.js。io.js的发展速度非常快，先是于2015年1月份发布了1.0版本，并且很快就达到了2.0版本，社区非常活跃。而最近io.js社区又宣布，这两个项目将合并到Node基金会下，并暂时由“Node.js和io.js核心技术团队联合监督”运营。本文将聊一聊Node.js项目的一些历史情况，与io.js项目之间的恩怨纠葛，他们将来的发展去向。希望能从历史的层面去了解这个开源项目在运营模式上是如何演变和发展的。

### Node.js项目的由来

自从JavaScript被Brendan Eich创造出来后，除了应用在浏览器中作为重要的补充外，人类从来就没有放弃过将JavaScript应用到服务端的想法。这些努力从livescript项目（1994年12月）开始，就从来没有停止过。如果你不知道livescript，那应该知道ASP中可以使用JScript语言（1996年），或者Rhino。但直到2009年，这些服务端JavaScript技术与同样应用在服务端的Java、PHP相比，显得相对失色。

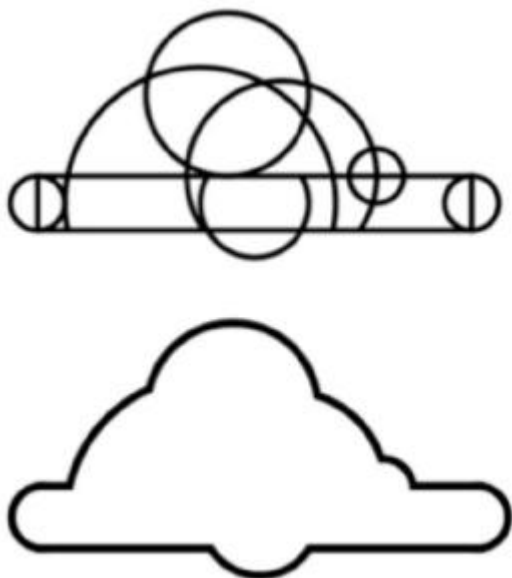
谈到Node.js的由来，不可避免要聊到它的创始人Ryan Dahl。在2009年时，服务端JavaScript迎来了它的拐点，因为Ryan Dahl带来了Node.js，在那之后Node.js将服务端JavaScript带入了新的境地，大量的JavaScript在GitHub上被贡献出来，大量的JavaScript模块出现，出现了真正的繁荣。

Node.js不是凭空出现的项目，也不是某个Web前端工程师为了完成将JavaScript应用到服务端的理想而在实验室里捣鼓出来的。它的出现主要归功于Ryan Dahl历时多年的研究，以及一个恰到好处的节点。2008年V8随着Chrome浏览器的出世，JavaScript脚本语言的执行效率得到质的提升，这给Ryan Dahl带来新的启示，他原本的研究工作与V8之间碰撞出火花，于是带来了一个基于事件的高性能Web服务器。



上图为Node.js创始人Ryan Dahl。

Ryan Dahl的经历比较奇特，他并非科班出身的开发者，在2004年的时候他还在纽约的罗彻斯特大学数学系读博士，期间有研究一些分形、分类以及p-adic分析，这些都跟开源和编程没啥关系。2006年，也许是厌倦了读博的无聊，他产生了『世界那么大，我想去看看』的念头，做出了退学的决定，然后一个人来到智利的Valparaiso小镇。那时候他尚不知道找一个什么样的工作来糊口，期间他曾熬夜做了一些不切实际的研究，如如何通过云进行通信。下面是这个阶段他产出的中间产物，与后来苹果发布的iCloud似乎有那么点相似。



从那起，Ryan Dahl不知道是否因为生活的关系，他开始学习网站开发了，走上了码农的道路。那时候Ruby on Rails很火，他也不例外的学习了它。从那时候开始，Ryan Dahl的生活方式就是接项目，然后去客户的地方工作，在他眼中，拿工资和上班其实就是去那里旅行。此后他去过很多地方，如阿根廷的布宜诺斯艾利斯、德国的科隆、奥地利的维也纳。

Ryan Dahl经过两年的工作后，成为了高性能Web服务器的专家，从接开发应用到变成专门帮客户解

决性能问题的专家。期间他开始写一些开源项目帮助客户解决Web服务器的高并发性能问题，尝试过的语言有Ruby、C、Lua。当然这些尝试都最终失败了，只有其中通过C写的HTTP服务库libebb项目略有起色，基本上算作libuv的前身。这些失败各有各的原因，Ruby因为虚拟机性能太烂而无法解决根本问题，C代码的性能高，但是让业务通过C进行开发显然是不太现实的事情，Lua则是已有的同步I/O导致无法发挥性能优势。虽然经历了失败，但Ryan Dahl大致的感觉到了解决问题的关键是要通过事件驱动和异步I/O来达成目的。

在他快绝望的时候，V8引擎来了。V8满足他关于高性能Web服务器的想象：

没有历史包袱，没有同步I/O。不会出现一个同步I/O导致事件循环性能急剧降低的情况。

V8性能足够好，远远比Python、Ruby等其他脚本语言的引擎快。

JavaScript语言的闭包特性非常方便，比C中的回调函数好用。

于是在2009年的2月，按新的想法他提交了项目的第一行代码，这个项目的名字最终被定名为“node”。

2009年5月，Ryan Dahl正式向外界宣布他做的这个项目。2009年底，Ryan Dahl在柏林举行的JSConf EU会议上发表关于Node.js的演讲，之后Node.js逐渐流行于世。

以上就是Node.js项目的由来，是一个专注于实现高性能Web服务器优化的专家，几经探索，几经挫折后，遇到V8而诞生的项目。

Node.js项目的组织架构和管理模式

Node.js随着JSConf EU会议等形式的宣传下，一家位于硅谷的创业公司注意到了该项目。这家公司就是Joyent，主要从事云计算和数据分析等。Joyent意识到Node.js项目的价值，决定赞助这个项目。

Ryan Dahl于2010年加入该公司，全职负责Node.js项目的开发。此时Node.js项目进入了它生命历程里的第二个阶段：从个人项目变成一个公司组织下的项目。

这个阶段可以从2010年Ryan Dahl加入Joyent开始到2014年底Mikeal Rogers发起Node Forward结束，Node的版本也发展到了v0.11。这个时期，IT业中的大多数企业都关注过Node.js项目，如微软甚至对于Node.js对Windows的移植方面做过重要的贡献。

这个时期可以的组织架构和管理模式可以总结为“Gatekeeper + Joyent”模式。

Gatekeeper的身份类似于项目的技术负责人，对技术方向的把握是有绝对权威。历任的Gatekeeper为：Ryan Dahl、Isaac Z. Schlueter、Timothy J Fontaine，均是在Node.js社区具有很高威望的贡献者。项目的法律方面则由Joyent负责，Joyent注册了“Node.js”这个商标，使用其相关内容需要得到法律授权（如笔者《深入浅出Node.js》上使用了Node.js的Logo，当时是通过邮件的形式得到过授权）。技术方面除了Gatekeeper外，还有部分core contributor。core contributor除了贡献重要feature外，帮助项目进行日常的patch提交处理，协助review代码和合并代码。项目中知名的core contributor有Ben Noordhuis，Bert Belder、Fedor Indutny、Trevor Norris、Nathan Rajlich等，这些人大多来自Joyent公司之外，他们有各自负责的重要模块。Gatekeeper除了要做core contributor的事情外，还要决定版本的发布等日常事情。

Node.js成为Joyent公司的项目后，Joyent公司对该项目的贡献非常大，也没有过多的干涉Node.js社区的发展，还投入了较多资源发展它，如Ryan Dahl、Isaac Z. Schlueter、Timothy J Fontaine等都是Joyent的全职员工。

## Node.js社区的分裂

“Gatekeeper + Joyent”模式运作到2013年的时候都还工作良好，蜜月期大概中止于第二任Gatekeeper Isaac Z. Schlueter离开Joyent自行创建npm inc.公司时期。前两任Gatekeeper期间，Node.js的版本迭代都保持了较高的频率，大约每个月会发布一个小版本。在Isaac Z. Schlueter卸任Gatekeeper之后，Node.js的贡献频率开始下降，主要的代码提交主要来自社区的提交，代码的版本下降到三个月才能发布一个小版本。社区一直期待的1.0版本迟迟不能发布。这个时期Node.js属于非常活跃的时期，但是对于Node.js内核而言却进展缓慢。技术方向上似乎是有些不明朗，一方面期待内核稳定下来，一方面又不能满足社区对新feature的渴望（如ES6的特性迟迟无法引入）。

第三任的Gatekeeper Timothy J Fontaine本人也意识到这个问题。从他上任开始，主要的工作方向就是解决该问题。他主要工作是Node on the road活动，通过一系列活动来向一些大企业用户获取他们使用Node.js的反馈。通过一些调研，他做了个决定，取消了贡献者的CLA签证，让任何人可以贡献代码。

尽管Timothy J Fontaine的做法对Node.js本身是好的，但是事情没有得到更好的改善。这时候Node.js项目对社区贡献的patch处理速度已经非常缓慢，经常活跃的core contributor只有Fedor Indutny、Trevor Norris。另外还发生了人称代词的事件，导致Node.js/libuv项目中非常重要的贡献者Ben Noordhuis离开core contributor列表，这件事情被上升到道德层面，迎来了不少人的谩骂。其中Joyent的前任CEO甚至还致信表示如果是他的员工，会进行开除处理。这致使Node.js项目的活跃度更低。Node.js的进展缓慢甚至让社区的知名geek TJ Holowaychuk都选择离开Node.js而投入Go语言的怀抱。

可以总结这个时期是“Gatekeeper + Joyent”模式的末期。Joyent对于项目的不作为和其他层面对社区其他成员的干预，导致项目进展十分缓慢，用蜗牛的速度来形容一点也不为过。尽管Timothy J Fontaine试图挽回些什么，也有一些行为来试图重新激活这个项目的活力，但是已经为时已晚。

这时一个社区里非常有威望的人出现了，他就是Mikeal Rogers。Mikeal Rogers的威望不是建立在他对Node.js项目代码的贡献上，他的威望主要来自于request模块和JSConf会议。其中JSConf是JavaScript社区最顶级的会议，他是主要发起人。

在2014年8月，以Mikeal Rogers为首，几个重要core contributor一起发起了一个叫做“Node forward”的组织。该组织致力于发起一个由社区自己驱动来提升Node、JavaScript和整个生态的项目。

“Node forward”可以视作是io.js的前身。这些core contributor们在“Node forward”上工作了一段时间，后来因为可能涉及到Node这个商标问题，Fedor Indutny愤而fork了Node.js，改名为io.js，宣告了Node.js社区的正式分裂。

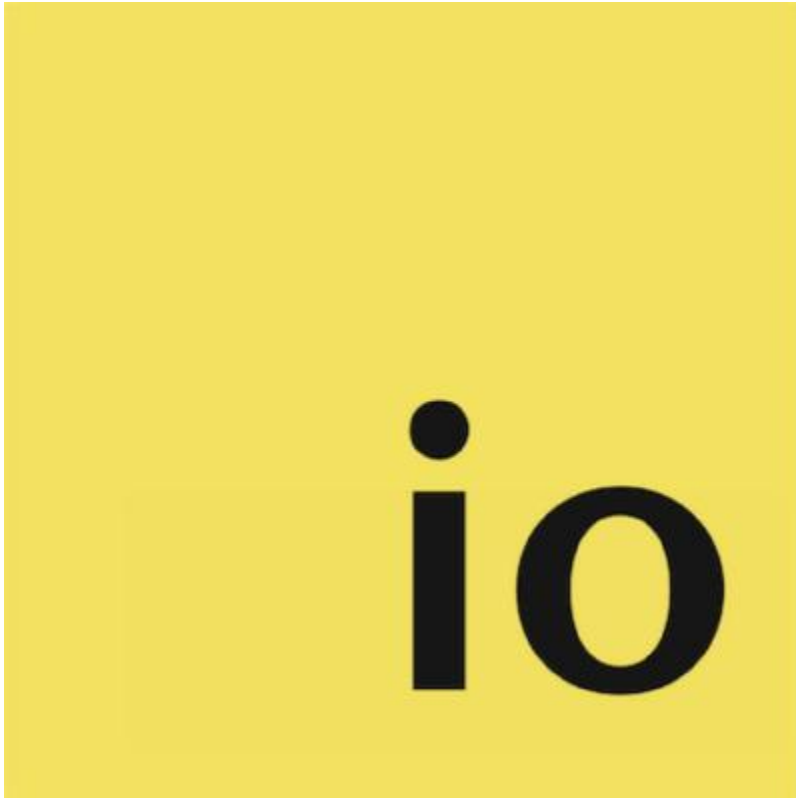
简单点来说这件事情主要在于社区贡献者们对于Joyent公司的不满，导致这些主要贡献者们想通过一个更开放的模式进行协作。复杂点来说这是公司开源项目管理模式的问题所在，当社区方向和公司方



向一致时，必然对大家都有好处，形同蜜月期，但当两者步骤不一致时，分歧则会暴露出来。这点在Node.js项目的后期表现得极为明显，社区觉得项目进展缓慢，而Joyent公司的管理层则认为他稳定可靠。

### io.js与Node.js advisory board

在“Node Forward”的进展期间，社区成员们一起沟通出了一个基本的开放的管理模式。这个模式在io.js期间得到体现。



io.js的开放管理模式主要体现在以下方面：

不再有Gatekeeper。取而代之的是TC（Technical Committee），也就是技术委员会。技术委员会基本上是由那些有很多代码贡献的core contributor组成，他们来决定技术的方向、项目管理和流程、贡献的原则、管理附加的合作者等。当有分歧产生时（如引入feature），采用投票的方式来决定，遵循少数服从多数的简单原则。基本上原来由一个人担任的Gatekeeper现在由一个技术委员会来执行。如果要添加一个新成员为TC成员，需要由一位现任的TC成员提议。每个公司在TC中的成员不能超过总成员的1/3。

引入Collaborators。代码仓库的维护不仅仅局限在几个core contributor手中，而是引入Collaborators，也就是合作者。可以理解为有了更多的core contributor。

TC会议。之前的沟通方式主要是分布式的，大家通过GitHub的issue列表进行沟通。这种模式容易堆积问题，社区的意见被接受和得到处理取决于core contributor的情况。io.js会每周举行TC会议，会议的内容主要就issue讨论、解决问题、工作进展等。会议通过Google Hangout远程进行，由TC赞同的委任主席主持。会议视频会发布在YouTube上，会议记录会提交为文档放在代码仓库中。

成立工作组。在项目中成立一些细分的工作组，工作组负责细分方向上的工作推进。

io.js项目从fork之后，于2015-01-14发布了v1.0.0版本。自此io.js一发不可收拾，以周为单位发布新的版本，目前已经发布到2.0.2。io.js项目与Node.js的不同在行为上主要体现在以下方面：

新功能的激进。io.js尽管在架构层面依然保持着Node.js的样子（由Ryan Dahl时确立），但是对于ECMAScript 6持拥抱态度。过去在Node.js中需要通过flag来启用的新功能，io.js中不再需要这些flag。当然不用flag的前提是V8觉得这个feature已经稳定的情况下。一旦最新的Chrome采用了新版本的V8，io.js保持很快的跟进速度。

版本迭代。io.js保持了较高频率的迭代，以底层API改变作为大版本的划分，但对于小的改进，保持每周一个版本的频率。只要是改进，io.js项目的TC和Collaborators都非常欢迎，大到具体feature或bug，小到文档改进都可以被接受，并很快放出版本。

issue反馈。Node.js的重要的贡献者们都在io.js上工作，Node.js和io.js项目的问题反馈速度几乎一致，但是问题处理速度上面io.js以迅捷著称，基本在2-3天内必然有响应，而Node.js则需要1个礼拜才有回复。

基本上而言原本应该属于Node.js项目的活力现在都在io.js项目这里。如果没有其他事情的发生，io.js可以算作社区驱动开源项目的成功案例了。

当然，尽管在Node.js这边进展缓慢，但Joyent方面还是做出了他们的努力。在“Node Forward”讨论期间，Joyent成立了临时的Node.js顾问委员会<https://nodejs.org/about/advisory-board/>。顾问委员会的主要目标与“Node Forward”的想法比较类似，想借助顾问委员会的形式来产出打造一个更加开放的管理模式，以找到办法来平衡所有成员的需要，为各方提供一个平台来投入资源到Node.js项目。

顾问委员会中邀请了很多重要的Contributor和一些Node.js重度用户的参与。开了几次会议来进行探讨和制定新的管理模式。于是就出现了一边是io.js如火如荼发布版本，Joyent这边则是开会讨论的情况。顾问委员会调研了IBM（Eclipse）、Linux基金会、Apache等，决定成立Node.js基金会的形式。

## io.js与Node.js基金会

时间来到2015年1月，临时委员会正式发布通告决定将Node.js项目迁移到基金会，并决定跟io.js之间进行和解。简单点来说Node.js方面除了版本的进展比较缓慢外，确实是在制定一个新的模式来确保Node.js项目的下一步发展，Joyent公司本着开放的原则，也做出相当大的让步，保持着较为和谐的状态。

然而io.js动作太快，代码的进展程度远远快于Node.js项目，和解的讨论从2月开始讨论，到5月才做出决定。这时io.js已经发布了它的2.0版本。

最终的结论是Node.js项目和io.js项目都将加入Node.js基金会。Node.js基金会的模式与io.js较为相似，但是更为健全。Mikeal Rogers在他的一篇名为《Growing Up》的文章中提到io.js项目需要一个基金会的原因。

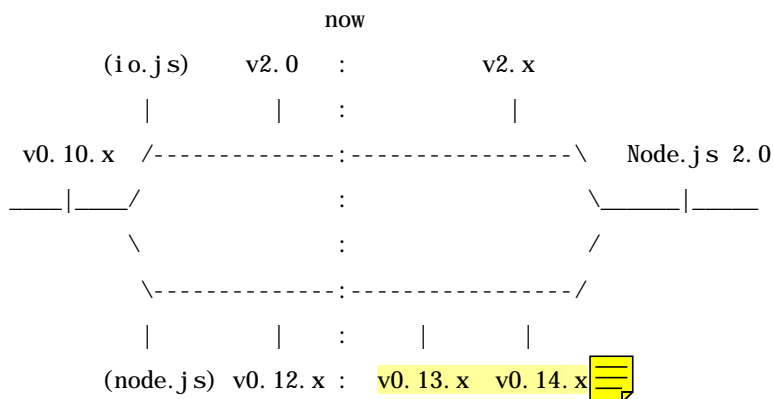
io.js项目在技术方面的成熟度显然要比最初的Gatekeeper时代要更为先进，给予贡献者更多的管理权利。然而在市场和法律方面，还略显幼稚。最终无论是顾问委员会，还是io.js都选定以基金会的形式存在。这个基金会参考Linux基金会的形式，由董事会和技术委员会组成，董事会负责市场和法律方面的事务，技术委员会负责技术方向。

就像《三国演义》所述：天下大势，合久必分，分久必合。Node.js项目也从Joyent公司的怀里走出来，成长为基金会的形式，进入这个项目生命周期里第三个阶段。

## 后续

从io.js的分裂到Node.js基金会，从外人看起来似乎如一场闹剧一般，然而这个过程中可以看到一个开源项目自身的成长。尽管io.js将归于Node.js基金会，像一个离家出走的孩子又回家一般，它的出走可能要被人忘记，但从当初的出发点来说，这场战役，io.js其实是赢家。穷则思变、不破不立是对Joyent较为恰当的形容。如果Joyent提前能相当这些，则不会有社区分裂的事情发生。

Node.js处于停滞状态的开发和io.js的活跃情况之间，目前免不了大量的Merge工作。作为和解的条件之一，Node.js基金会之后Node版本的发布将基于目前io.js的进展来进行。后续的合并工作示意如下：



在未完成合并之前，io.js会继续保持发布。Node.js的下个大版本跨过1.0，直接到2.0。

io.js项目的TC将被邀请加入Node.js基金会的TC，毕竟两者在技术管理方面达成了一致。基金会将在黄金和白银会员中选举出董事、技术委员会成员中选举出技术委员主席。

对于成为Node.js基金会成员方面，企业可以通过赞助的方式注册成为会员。

## 总结

一个开源项目成长起来之后，就不再是当初创始人个人维护的那个样子了。Node.js项目的发展可以说展现了一个开源项目是如何成长蜕变成成熟项目的。当然我们现在说Node.js基金会是成功的还为时尚早，但是祝福它。

## 参考文档

<https://github.com/joyent/node/issues/9295> (<https://github.com/joyent/node/issues/9295>)

<https://github.com/iojs/io.js/issues/978> (<https://github.com/iojs/io.js/issues/978>)

<https://github.com/iojs/io.js/issues/1336> (<https://github.com/iojs/io.js/issues/1336>)

<https://github.com/iojs/io.js/issues/1416> (<https://github.com/iojs/io.js/issues/1416>)

<https://github.com/iojs/io.js/labels/meta> (<https://github.com/iojs/io.js/labels/meta>)

<http://blog.nodejs.org/2015/05/08/transitions/> (<http://blog.nodejs.org/2015/05/08/transitions/>)

<http://blog.nodejs.org/2015/05/08/next-chapter/> (<http://blog.nodejs.org/2015/05/08/next-chapter/>)

<https://github.com/iojs/io.js/issues/1664> (<https://github.com/iojs/io.js/issues/1664>)

<http://tinyclouds.org/nodeconf2012.pdf> (<http://tinyclouds.org/nodeconf2012.pdf>)

<https://www.joyent.com/blog/introducing-the-nodejs-foundation> (<https://www.joyent.com/blog/introducing-the-nodejs-foundation>)

<http://blog.nodejs.org/2015/05/15/node-leaders-are-building-an-open-foundation/>

(<http://blog.nodejs.org/2015/05/15/node-leaders-are-building-an-open-foundation/>)

<https://medium.com/node-js-javascript/growing-up-27d6cc8b7c53> (<https://medium.com/node-js-javascript/growing-up-27d6cc8b7c53>)

9 回复



(/user/Pana) Pana (/user/Pana) 1楼•2 年前

原文地址：<http://www.infoq.com/cn/articles/node-js-and-io-js> (<http://www.infoq.com/cn/articles/node-js-and-io-js>)



(/user/qianjiahao) qianjiahao (/user/qianjiahao) 2楼•2 年前

看完好过瘾~



(/user/pockry) pockry (/user/pockry) 3楼•2 年前

这篇文章是InfoQ的版权文章，你全文转载也就算了，为啥还给了精华.....



(/user/dd1994) dd1994 (/user/dd1994) 4楼•2 年前

Ryan Dahl经过两年的工作后，成为了高性能Web服务器的专家

一个业余选手，随便接项目，经过短短两年就成为专家，果然天才的想法是我们正常人无法理解的  
orz orz



(/user/Pana) Pana (/user/Pana) 5楼•2 年前


@dd1994 (/user/dd1994) 在计算机行业，天才和普通人能够差出非常大的距离



(/user/NGFIX) NGFIX (/user/NGFIX) 6楼•2 年前

好文啊~简史一般



 (/user/Pana) Pana (/user/Pana) 7楼•2 年前


@pockry (/user/pockry) 标题已经说明是朴灵大大的文章

---

 (/user/knightuniverse) knightuniverse (/user/knightuniverse) 8楼•2 年前

一口气看完了

---

 (/user/zmzkkk) zmzkkk (/user/zmzkkk) 9楼•10 个月前

过有后续的 5.0 版本 发展历史呢？