SYMBIO

2018,我们来谈谈Node.js

Victor Tang

目 录

- Node.js
  - 历 史
  - 槽点
  - 选择
  - 架构平衡
- 当前技术栈
  - 应用场景
  - Web框架
  - 跨平台
- 未 来

SYMBIO

前言

Node.js 是一个非常新兴的开发工具，它诞生自 2009 年，是有史以来发展最快的开发工具，没有之一。在这短短的几年间，我们看到了 Node.js 从当初的一无所有到如今的飞速发展，这是没有任何其他开发工具能够媲美的。

Node.js不是Javascript应用，Node.js采用C++语言编写而成，是一个Javascript的运行环境。下面我们看看官方的定义：Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行时。 Node.js 使用高效、轻量级的事件驱动、非阻塞 I/O 模型。它的包生态系统，npm，是目前世界上最大的开源库生态系统。

历 史

1. 前传
2. 歪果仁Ryan Dahl(混沌期)
3. 成长期
4. io.js(分裂期)
5. Node.js基金会(飞速发展期)

历史
前传

95年Netscape设计出JavaScript后在浏览器上大获成功,同期有Netscape的JavaScript、微软的JScript以及CEnvi的ScriptEase三足鼎立的格局。

97年，在ECMA（欧洲计算机制造商协会）的协调下，确定为统一标准：ECMA-262，我们今天所说的javascript一般是指ECMAScript。

微软通过IE击败了Netscape后一统桌面浏览器，结果几年时间，浏览器毫无进步。（2001年出的古老的IE 6到今天仍然有人在使用！）

先是Mozilla借助已壮烈牺牲的Netscape遗产在2002年推出了Firefox浏览器，紧接着Apple于2003年在开源的KHTML浏览器的基础上推出了WebKit内核的Safari浏览器，不过仅限于Mac平台。

随后，Google也开始创建自家的浏览器。他们也看中了WebKit内核，于是基于WebKit内核推出了Chrome浏览器。

历史
前传

Chrome浏览器是跨Windows和Mac平台的，并且，Google认为要运行现代Web应用，浏览器必须有一个性能非常强劲的JavaScript引擎，于是Google自己开发了一个高性能JavaScript引擎，名字叫V8，以BSD许可证开源。

研究公司StatCounter 2017年6月浏览器调查报告显示，世界范围内的桌面浏览器市场占有率Chrome占比为63.23%，国内桌面浏览器的市场占有率Chrome占比为59.13%，国内手机端浏览器的市场占有率Chrome占比为66.52%。

而研究公司StatCounter 2002年4月浏览器调查报告显示，IE的市场份额高达96.6%（IE 6），现代浏览器大战让微软的IE浏览器远远地落后了，IE浏览器从此与主流移动端设备绝缘。

历史

混沌期

浏览器大战和Node.js有何关系？话说有个叫Ryan Dahl的歪果仁，2004年他还在纽约的罗彻斯特大学数学系读博士，2006年退学，经过两年的工作后成为了高性能Web服务器的专家，当时他的工作是用C/C++写高性能Web服务。对于高性能，异步IO、事件驱动是基本原则，但是用C/C++写就太痛苦了。

于是这位大神开始设想用高级语言开发Web服务。他评估了很多种高级语言，几经探索，几经挫折后，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。

V8性能足够好。

JavaScript语言的闭包特性非常方便，比C中的回调函数好用。

在2009年，Ryan Dahl正式推出了基于JavaScript语言和V8引擎的开源Web服务器项目，命名为Node.js。

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。

历史

成长期

2010年年底，Node.js获得云计算服务商Joyent资助，创始人Ryan Dahl加入Joyent全职负责Node.js的发展。

2011年7月，Node.js在微软的支持下发布Windows版本。

2011年年底，Node.js 的核心用户 艾萨克·施吕特（Isaac Z. Schlueter）开发出奠定了 Node.js 如今地位的重要工具--npm。connect, express, socket.io 等库的出现，CoffeeScript 的出现，以及以 Node.js 作为运行环境的 CLI 工具，如 less, UglifyJS, browserify, grunt 等的出现助推了Node.js的繁荣发展。

历 史
分裂期

因为Node.js是开源项目，虽然由社区推动，但幕后一直由Joyent公司资助。由于一群开发者对Joyent公司的策略不满，于2014年从Node.js项目fork出了io.js项目，决定单独发展，于2015-01-14发布了v1.0.0版本。

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、版本迭代频率较高、issue反馈迅捷。

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。

历 史

飞速发展期

2015年Joyent终于宣布成立Node.js的开源基金会，这标志着Joyent将交出Node.js的控制权，Node.js将进入一个全新的发展阶段。自此，将由 Joyent、IBM、Paypal、微软、Fidelity和Linux基金会等创始成员共同掌管Node.js开源项目。

IO.js 和 Node.js 在 Node.js 基金会下合并，同年底首个长期支持版本（LTS）Node.js v4.2.0发布。

NODE.JS基金会白金会员

SYMBIO

槽 点

1.版本帝
2.已无性能优势？
3.异步和回调地狱？

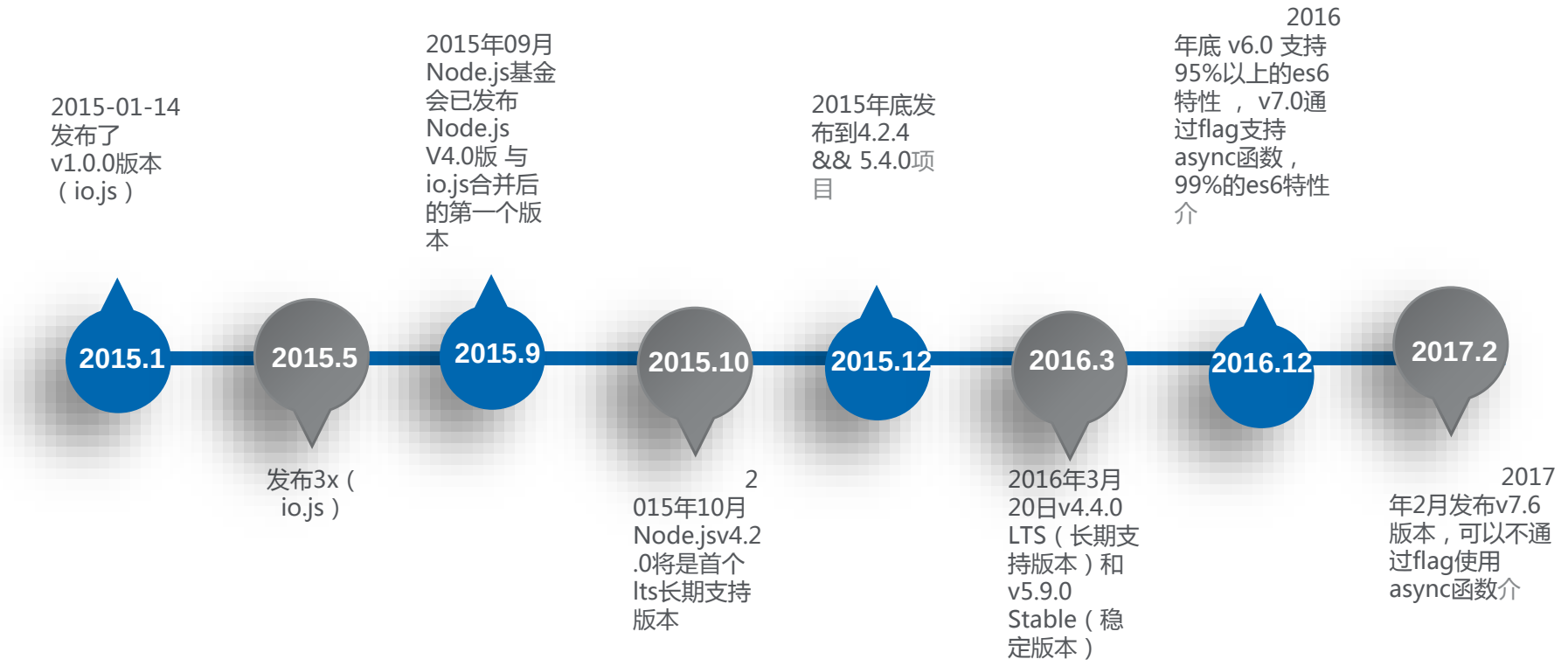
槽点一


版本帝
Chrome浏览器已经蹦到63版本了，是名副其实的版本帝，作为兄弟的Node.js也一样，1.0之前等了6年，而从1.0到8.0，只用了2年时间，这世界到底怎么了？

SYMBIO

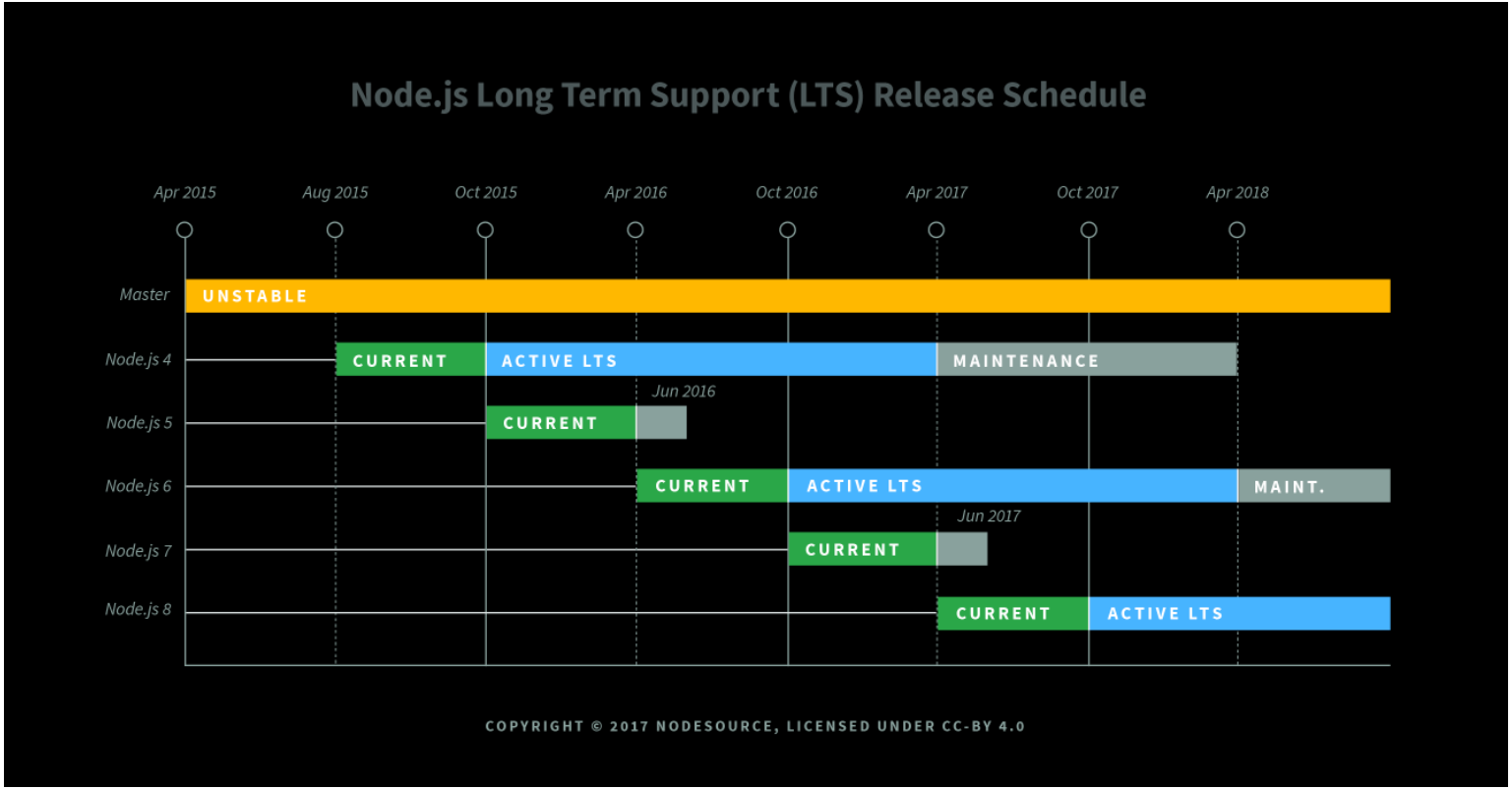
版本帝

从v0.1到0.12用了6年



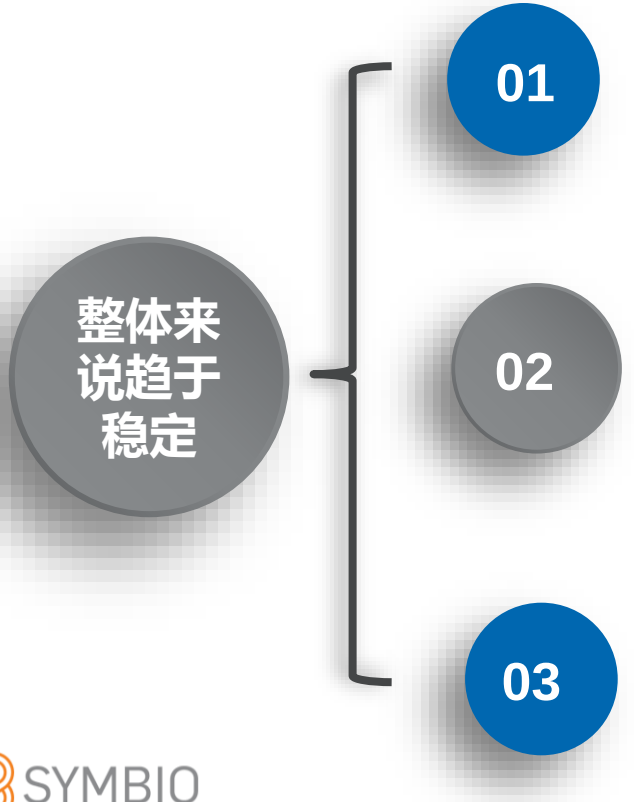
SYMBIO

版本帝


Node.js Long Term Support (LTS) Release Schedule
Apr 2015
Aug 2015
Oct 2015
Apr 2016
Oct 2016
Apr 2017
Oct 2017
Apr 2018
Master
UNSTABLE
Node.js 4
CURRENT
ACTIVE LTS
MAINTENANCE
Jun 2016
Node.js 5
CURRENT
Node.js 6
CURRENT
ACTIVE LTS
MAINT.
Jun 2017
Node.js 7
CURRENT
Node.js 8
CURRENT
ACTIVE LTS
COPYRIGHT © 2017 NODESOURCE, LICENSED UNDER CC-BY 4.0

SYMBIO

版本帝



成立了Node.js基金会，能够让Node.js在未来有更好的开源社区支持。

发布了LTS版本，意味着API稳定

快速发版本，很多人吐槽这个，其实换个角度看，这也是社区活跃的一个体现，但如果大家真的看CHANGELOG，其实都是小改进，而且是边边角角的改进，也就是说Node.js的core（核心）已经非常稳定了，可以大规模使用。

SYMBIO

槽点二

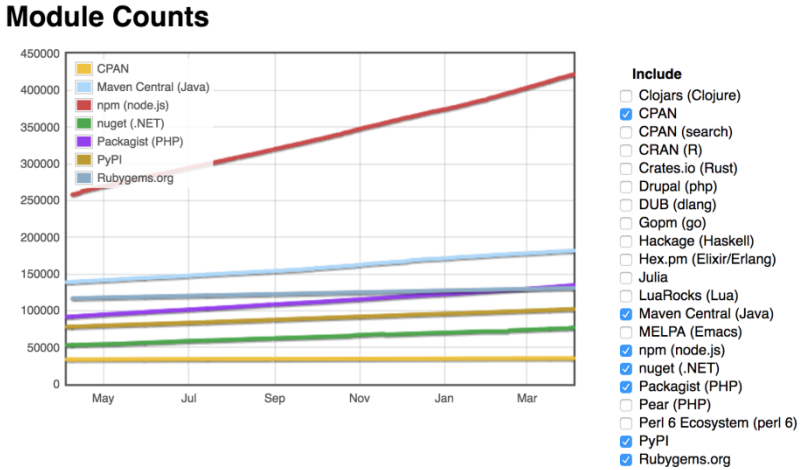

已无性能优势？
Node.js在2009年横空出世，可以说是纯异步获得高性能的功劳。所有语言几乎没有能够和它相比的，比如Java、PHP、Ruby都被啪啪的打脸。但是山一程，水一程，福祸相依，因为性能太出众，导致很多语言、编程模型上有更多探索，比如go语言产生、php里的swolo和vm改进等，大家似乎都以不支持异步为耻辱。后来的故事大家都知道了，性能都提到非常高，异步、IO问题已经没人再考虑，只是大家实现早晚而产生的性能差距而已。

SYMBIO

这其实是误读

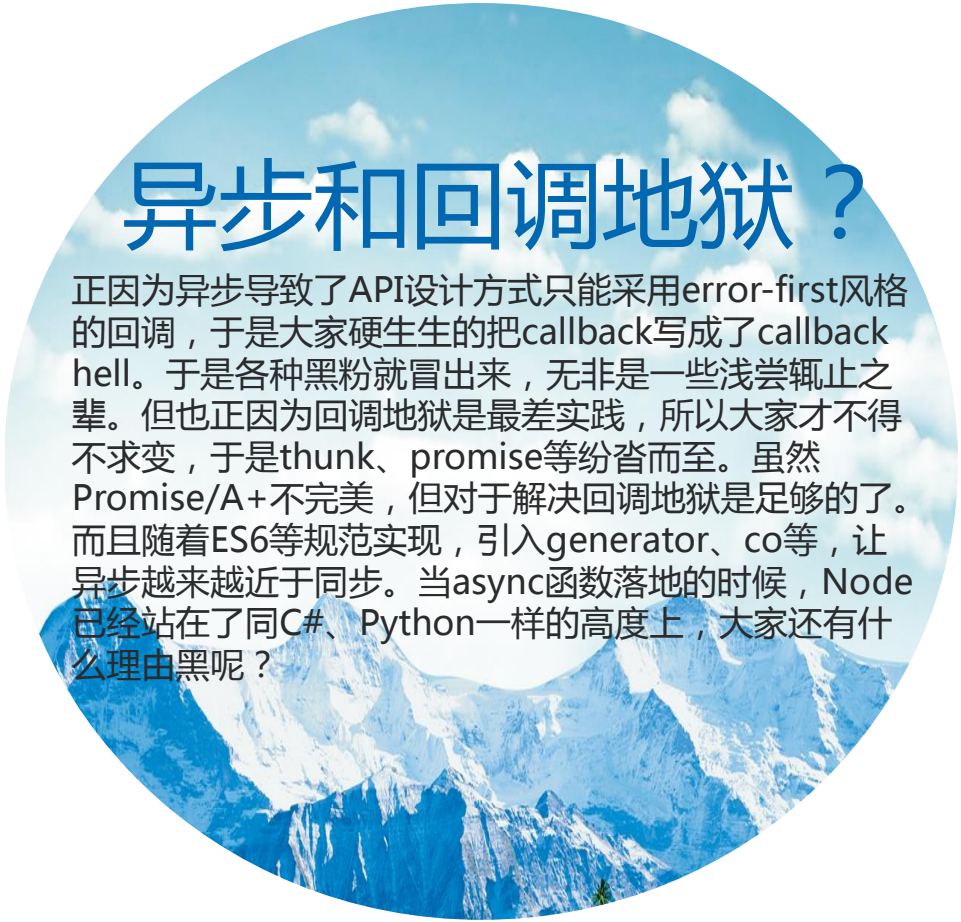
已无性能优势？

› 截止2017年2月，在npm上有超过45万个模块。npm是所有的开源的包管理里最强大的，并且Node.js的性能依然很好，且它有npm极其强大的生态，可谓性能与生态双剑合璧



SYMBIO

槽点三



异步和回调地狱？

正因为异步导致了API设计方式只能采用error-first风格的回调，于是大家硬生生的把callback写成了callback hell。于是各种黑粉就冒出来，无非是一些浅尝辄止之辈。但也正因为回调地狱是最差实践，所以大家才不得不求变，于是thunk、promise等纷沓而至。虽然Promise/A+不完美，但对于解决回调地狱是足够的了。而且随着ES6等规范实现，引入generator、co等，让异步越来越近于同步。当async函数落地的时候，Node已经站在了同C#、Python一样的高度上，大家还有什么理由黑呢？

SYMBIO

01 说明：Node.js API天生就是这样的

名称：callback

02 说明：参数的求值策略

名称：thunk

03 说明：最开始是Promise/A+规范，随后成为ES6标准

名称：promise

04 说明：ES6的一种生成器，用于计算，但tj想用做流程控制

名称：generator

05 generator用起来非常麻烦，故而tj写了co这个generator生成器，用法更简单

名称：co

06 原本计划进入es7规范，结果差一点，但好在v8实现了，所以node7就可以使用，无须等es7规范落地

名称：async函数

SYMBIO

选 择

1.学习成本

2.开发成本

可难可易
学习成本

- 可以采用面向过程
- 可以面向对象
- 可以函数式
- 甚至可以用各种编译器 coffee、typescript、babel（es）等

提供好的基础和包管理工具

开发成本

- 测试相关 tdd/bdd/测试覆盖率
- 规范化 standard、各种 lint、hint
- 构建相关 gulp、grunt、webpack，大量插件
- 生成器 yo等
- 包管理工具npm足够简单易用

架构平衡

1.在语言层面可以做，那语言层面做

2.如果语言层面搞不定，那就架构层面做

3.如果架构层面也解决不了……

1、在语言层面可以做，那语言层面做

架构平衡

已有大量 npm 上的模块（目前在 25.6 万个以上）

自己造轮子（站在海量包上 简单语法 npm = 快速）

使用 Node.js 里的 （ nan https://github.com/nodejs/nan ）自己包装 C/C++ 轮子

从上面看，绝大部分需求都可以满足了

SYMBIO

2、如果语言层面搞不定，那就架构层面做

架构平衡

业务边界、模块拆分、面向服务

MQ、RPC、cache

运维、监控、自动化

稍微解释一下，首先，架构与 Node.js 没直接关系。其次，架构师常用的东东有足够的 Node.js 模块支持，比如 MQ，像 Rabbitmq 有比较好的 Node 模块支持，RPC 里 Thrift、Grpc、Tchannel 支持的都不错，我们使用的 senecajs，Redis，ioredis 等软件，后面做 HA 都是一样的。

SYMBIO

3、如果架构层面也解决不了……

架构平衡

合适的场景用合适的东西。有很多东西是 Node.js 不擅长，又不在架构范畴里的，咋办？如实在不够，Java 补（严格点，应该叫其他语言补）

比如复杂 excel 生成

比如 apns 推送（Go 做其实也很好，不过除了我，没人能维护）

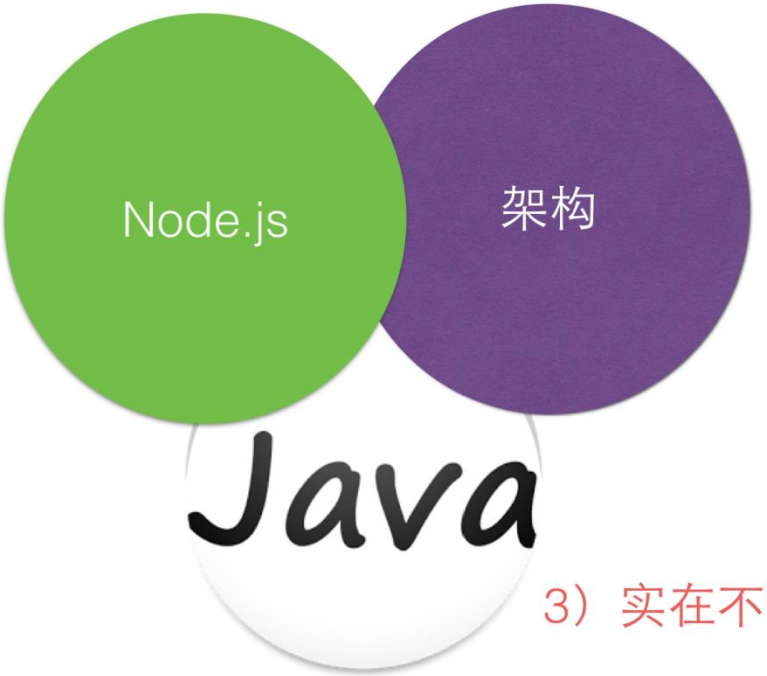
但凡是 Java 或其他语言里比较成熟的库，可以作为独立服务使用的，都可以做 Node.js 的支持。避免过多的时间用在造轮子上，影响开发进度。

架构平衡

架构平衡

1）在语言层面可以做，那语言层面做

2）如果语言层面搞不定，那就架构层面做



3）实在不够，java补

SYMBIO

应用场景

MEAN.JS

应用场景

（MEAN是一个Javascript平台的现代Web开发框架总称，它是MongoDB + Express +AngularJS + NodeJS 四个框架的第一个字母组合。它与传统LAMP一样是一种全套开发工具的简称。在2014和2015年喜欢讲这个，并且还有MEAN.js等框架，但今天已经过时，Node.js有了更多的应用场景。

SYMBIO

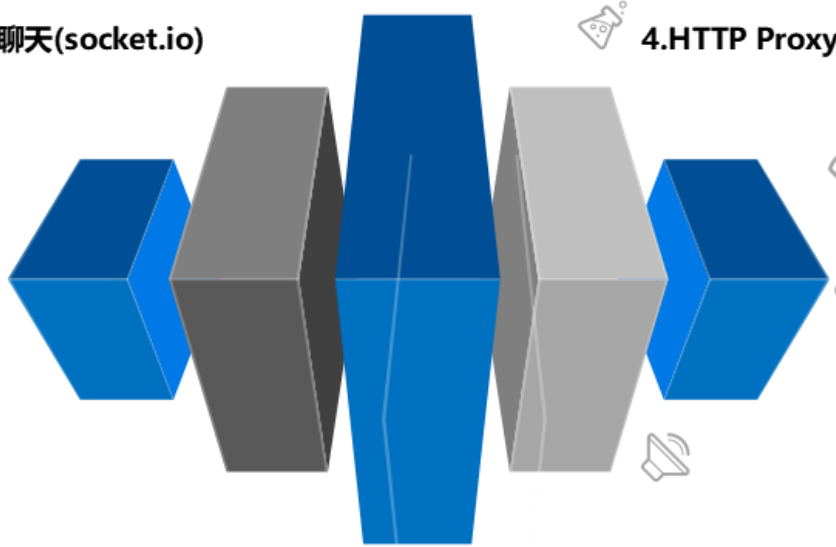
Node.js的若干使用场景

3.API（移动端，pc，h5）

2.im即时聊天(socket.io)

4.HTTP Proxy（淘宝、Qunar、腾讯、百度都有）

1.网站（如express/koa等）



5.前端构建工具(grunt/gulp/bower/webpack/fis3...)

6.写操作系统（NodeOS）

10.编辑器Atom、VSCode等

9.反向代理（比如anyproxy，node-http-proxy）

8.命令行工具（比如cordova、shell.js）

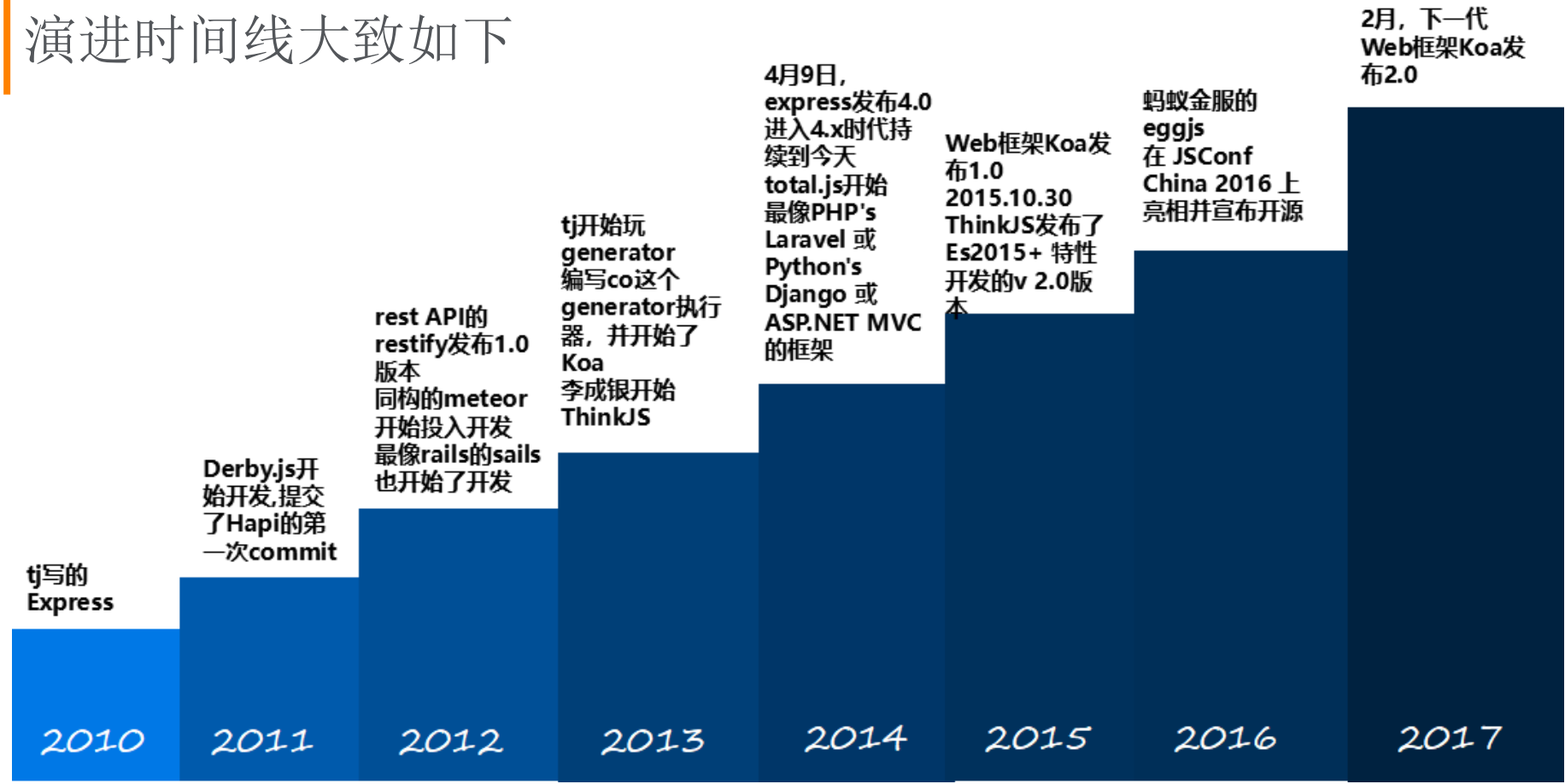
SYMBIO

WEB框架

1.演进时间线进行分类

2.根据框架的特性进行分类

演进时间线大致如下



SYMBIO

根据框架的特性进行分类

框架名称	特性	点评
Express	简单、实用，路由中间件等五脏俱全	最著名的Web框架
Derby.js && Meteor	同构	前后端都放到一起，模糊了开发便捷，看上去更简单，实际上上对开发来说要求更高
Sails、Total	面向其他语言，Ruby、PHP等	借鉴业界优秀实现，也是Node.js成熟的一个标志
MEAN.js	面向架构	类似于脚手架，又期望同构，结果只是蹭了热点
Hapi 和 Restfy	面向Api && 微服务	移动互联网时代Api的作用被放大，故而独立分类。尤其是对于微服务开发更是利器
ThinkJS	面向新特性	借鉴ThinkPHP，并慢慢走出自己的一条路，对于Async函数等新特性支持，无出其右
Koa	专注于异步流程改进	下一代Web框架

A Express

B Derby.js&& Meteor

C Sails,Total

D MEAN.js

E HapiRestfy

F Think.JS

G Koa

SYMBIO

SYMBIO

跨平台

1.Web/H5

2.移动端

3.PC端

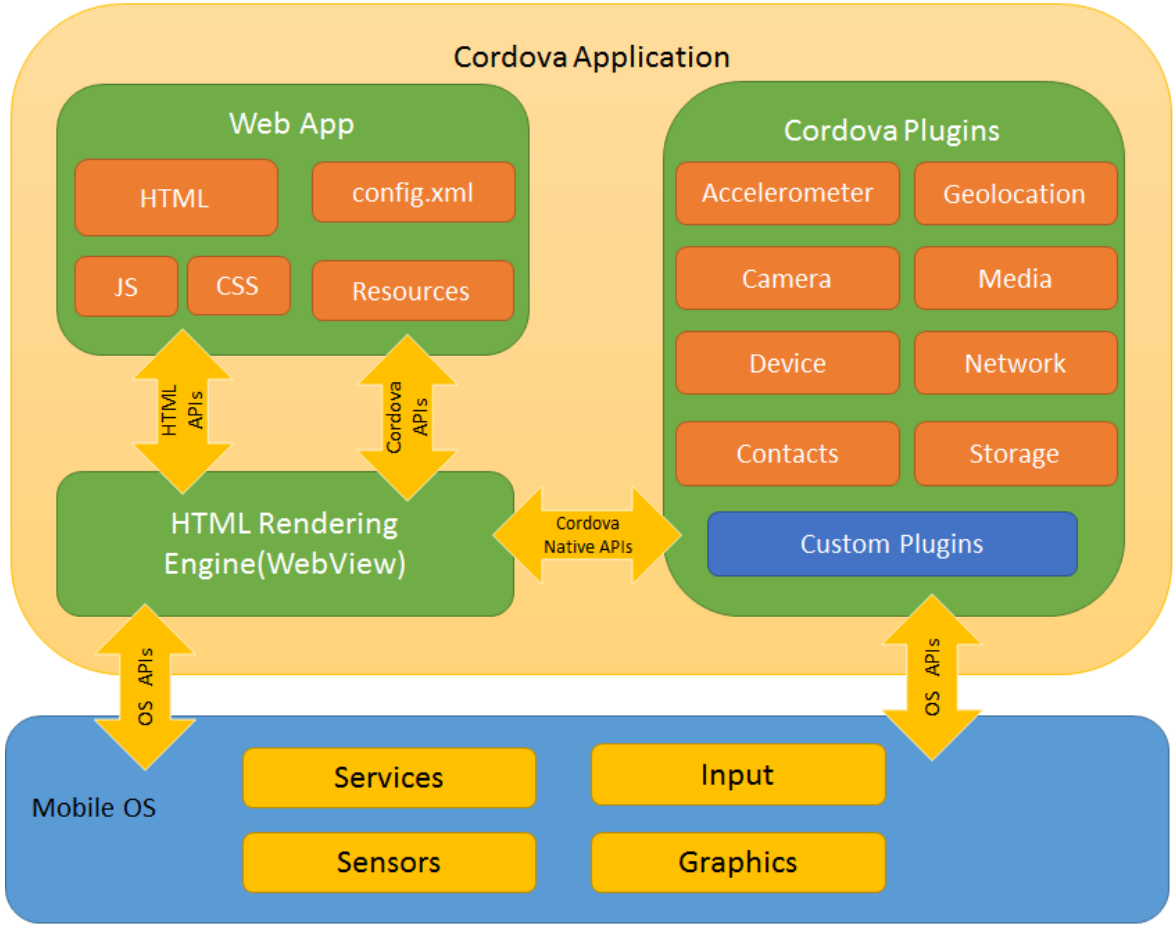
全栈 OR 全烂 ？

跨平台

SYMBIO

移动端：加壳

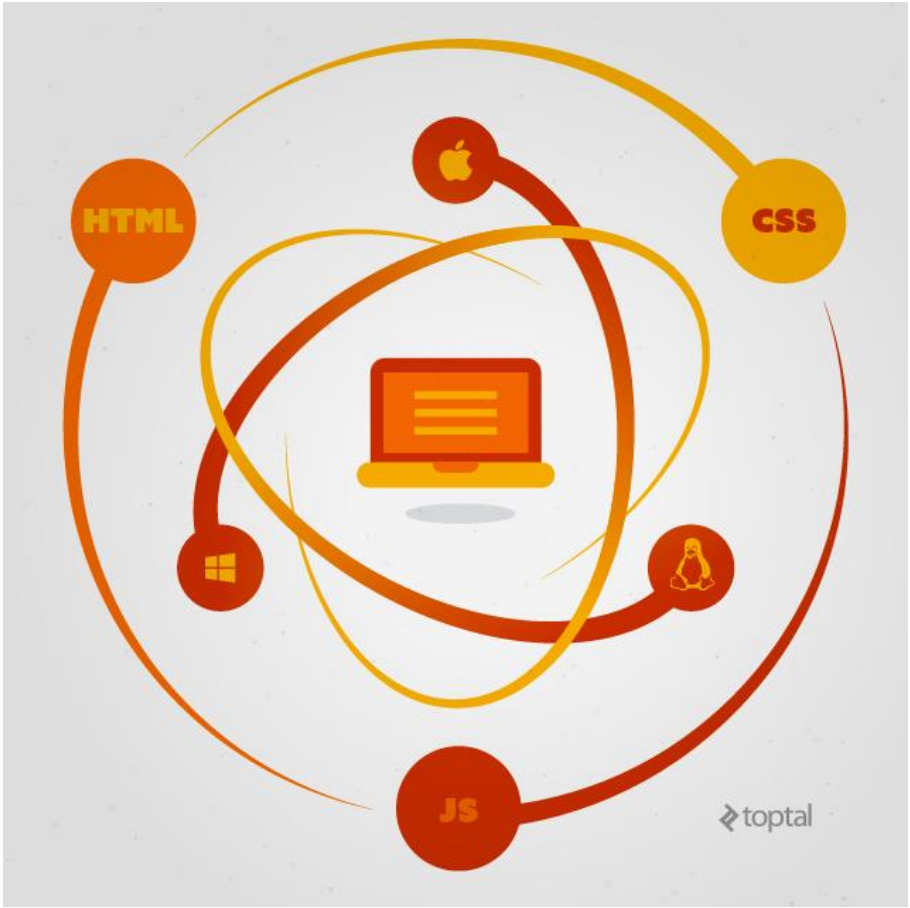
跨平台



SYMBIO

PC端：继续加壳

跨平台



SYMBIO

SYMBIO

未 来

未来

不该指望 「XXX 取代 XXX」、甚至执着于「 XXX 宇宙第一」

SYMBIO

未来

Node.js可能一个变革机遇，也可能是技术中更迭的一个过客，我们更相信它是变革机遇，拭目以待吧

SYMBIO

THE END 谢谢!

SYMBIO

推荐网址

参考

https://www.liaoxuefeng.com

https://i5ting.github.io/nodejs-fullstack/

https://cnodejs.org/

http://www.ruanyifeng.com