复杂系统的原理及其在 Node.js 系统开发中的应用

王韬 Wiredcraft 2021-05-23

内容

- JavaScript/Node.js的毁与誉
- 从复杂系统科学的视角来评估JavaScript/Node.js
  - 复杂系统基本原理
  - 如何应对环境复杂性
  - 如何应对环境不确定性
- 总结

Node.js诞生前

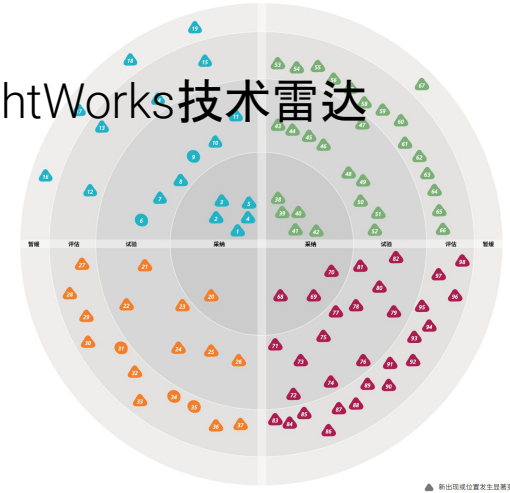
JavaScript从诞生到被鄙视

- 1995年JavaScript诞生
- 1997年ECMAScript 1
- 1998年ECMAScript 2
- 1999年ECMAScript 3
- ……
- 2008年废除ECMAScript 4
- 2008年Google发布Chrome浏览器和V8 JavaScript引擎

Node.js从诞生到快速流行

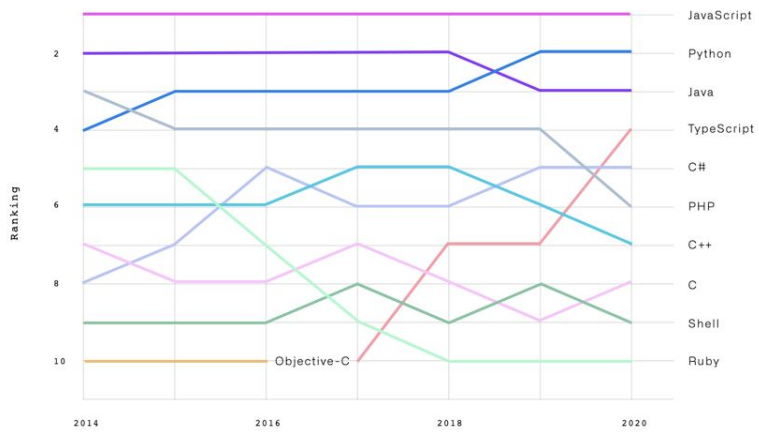
JavaScript开始快速演化并攻城掠地

- 2009年Node.js诞生
- 2009年ECMAScript 5
- 2011年ECMAScript 5.1
- 2011-2014年JavaScript和Node.js进入ThoughtWorks技术雷达
- 2015年ECMAScript 2015 (ES6)
- 2016年ECMAScript 2016
- 2017年ECMAScript 2017



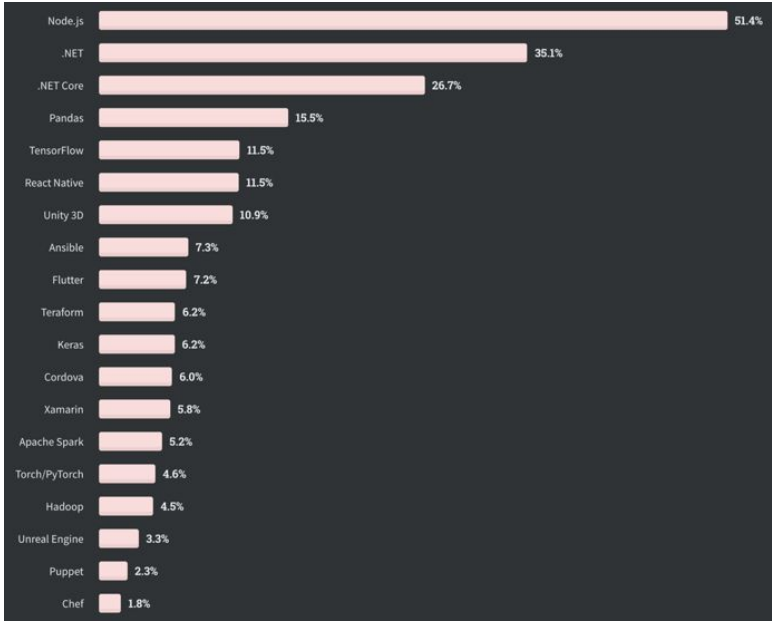
JavaScript如日中天

长期保持为GitHub上使用人数最多的语言



Node.js广泛使用

Stack Overflow 开发者报告中超过半数的被调查对象使用Node.js



什么样的应用适合JavaScript

Node.js是否被过度使用了

- 2007年Atwood定律：一切可以用JavaScript来写的应用，最终都将用JavaScript来写
- ……
- 2020年ThoughtWorks技术雷达警惕Node.js泛滥

直觉与分析

常见的决策依据

- 自己过去的经验
- 他人的经验和成功案例
- 标准化的框架 ISO/IEC 25010:2011, ……
- ……

小结

很多定性或者定量的分析框架都隐含着一些前提假设，这些前提假设在有些软件系统中并不成立。复杂系统科学会澄清这些假设在什么时候不成立，并提供了一些理解系统属性的其他框架。

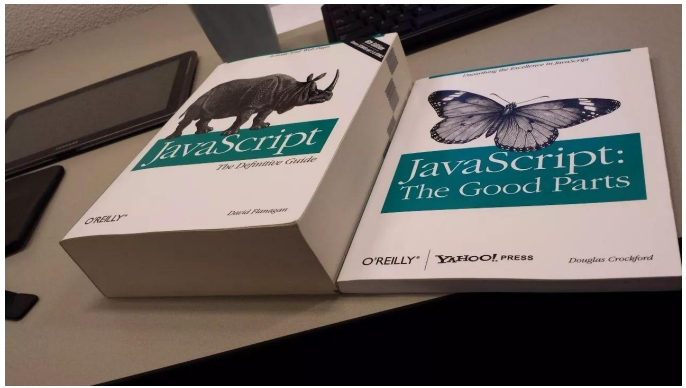
复杂系统的基本原理

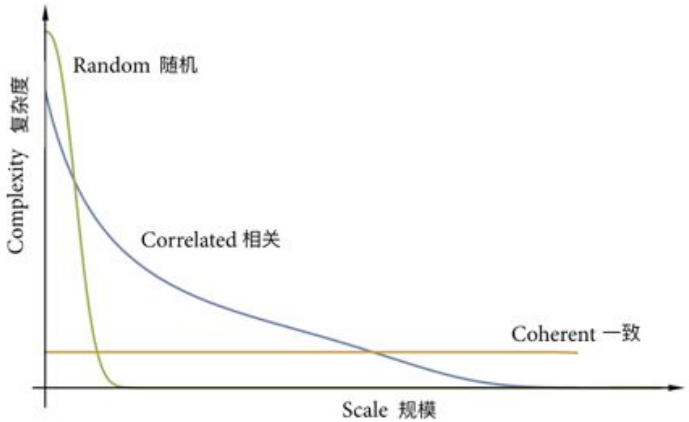
从复杂系统科学的角度看Node.js

- 量化系统复杂度
- 复杂系统常见的约束与应对策略
- 与环境复杂度匹配
- 管理系统复杂度
- 应对环境的不确定性

复杂度与规模和抽象层次有关

系统复杂度等于描述系统行为所需要的文字长度





适应性与效率不可兼得

系统整体复杂度为
各组成部分复杂度之和

- 整体的行为需要协调各组成部分的行为
- 各组成部分之间的相互关联依赖会限制各组成部分的行为复杂度
- 在能够完成任务的前提下，功能简单的语言更适合构建复杂系统
- 经过约束后的JavaScript比其他语言更简单，更适合构建复杂系统

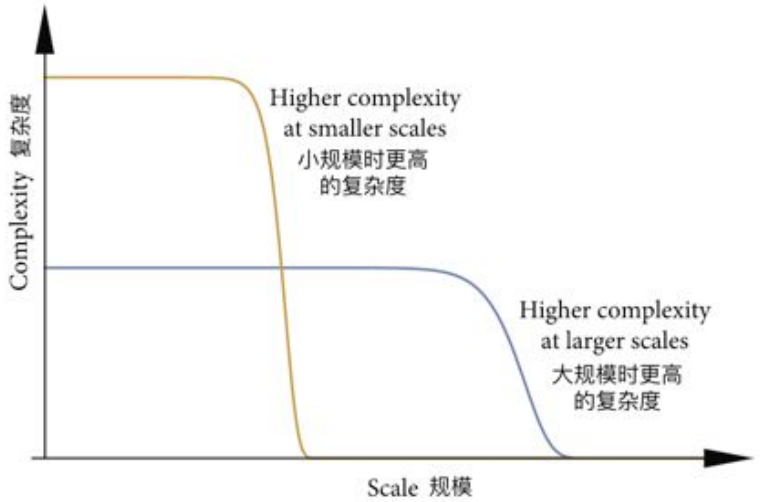
否则系统将无法有效应对环境中的某些场景

系统复杂度需要
大于或等于环境复杂度

- 能存活下来的系统不是某个单项最强的系统，而是最适应环境的系统
- IO密集型应用：Node.js标准库原生非阻塞IO
- 计算密集型应用: Node.js C++插件
- 小规模应用：JavaScript短平快绕过条条框框
- 大规模应用：TypeScript静态约束利于协作

否则系统将无法有效应对环境中的某些场景

系统在不同层面上的复杂度都需要和环境匹配



组件划分需要适应环境

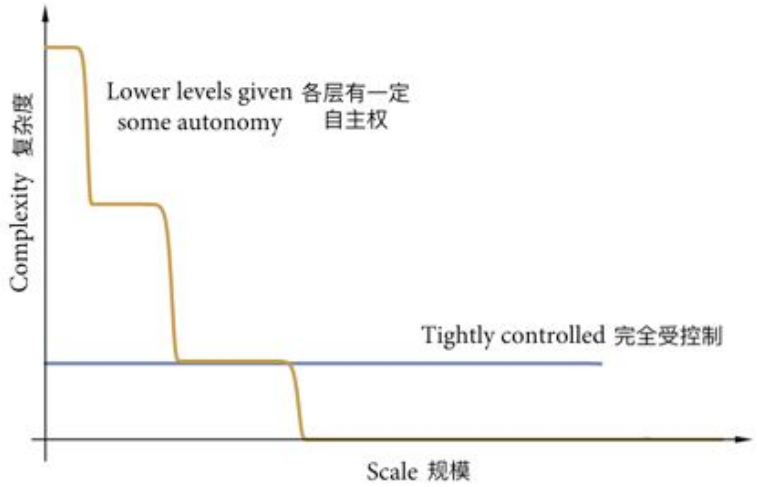
通过拆分组件来管理系统

- 微服务架构？
- 领域驱动设计？
- 需要多组件共同参与处理的任务怎么办?
- Conway定律:系统架构设计需要适应组织的沟通结构

层级间的关系需要适应环境

通过分层来管理系统

整洁架构
将稳定的高层抽象从易变的底层细节中分离出来



探索各种可能性，交流传播成功的方案

通过进化过程
来应对环境的不确定性

- 敏捷实践里的迭代回顾会议
- 跨团队知识交流会
- 多种框架express, koa, nest.js, egg.js, midway, …react, vue, angular, svelte, …
- 多种实验性语言特性TC39 Stage0 稻草人, Stage1 提议, Stage2 草案, Stage3 候选 => Stage4 完成

总结

- 系统复杂度等于描述系统行为所需要的文字长度
- 适应性与效率不可兼得
- 系统在不同层面上的复杂度都需要和环境匹配
- 可以通过拆分子系统或者系统分层来管理复杂度，但拆分和分层方式需要和环境匹配
- 通过进化过程来应对环境的不确定性
- JavaScript语言简单，Node.js生态强大，支持多种环境，且不断进化，适合用来构建多种复杂应用程序

参考资料

- The State of the Octoverse 2020, GitHub
- Stack Overflow Developer Survey 2020, Stack Overflow
- Software Architecture in Practice, 2012, 3rd Edition, Addison-Wesley
- An Introduction to Complex Systems Science and Its Applications, Complexity, vol. 2020, Article ID 6105872

总的来说

我们为全世界的优质品牌创造数字产品

员工福利

加入我们，你将获得的福利包括且不限于……


Flexible hours


Work from home


Parental leave


Friday lunch


Gym allowance


Personal growth

现在申请

加入我们的团队

与热情专注、追求卓越的伙伴们一起工作和成长。目前热招的岗位包括：

- 运维工程师
- 数据分析师
- QA 测试工程师
- 前端开发工程师
- 后端开发工程师