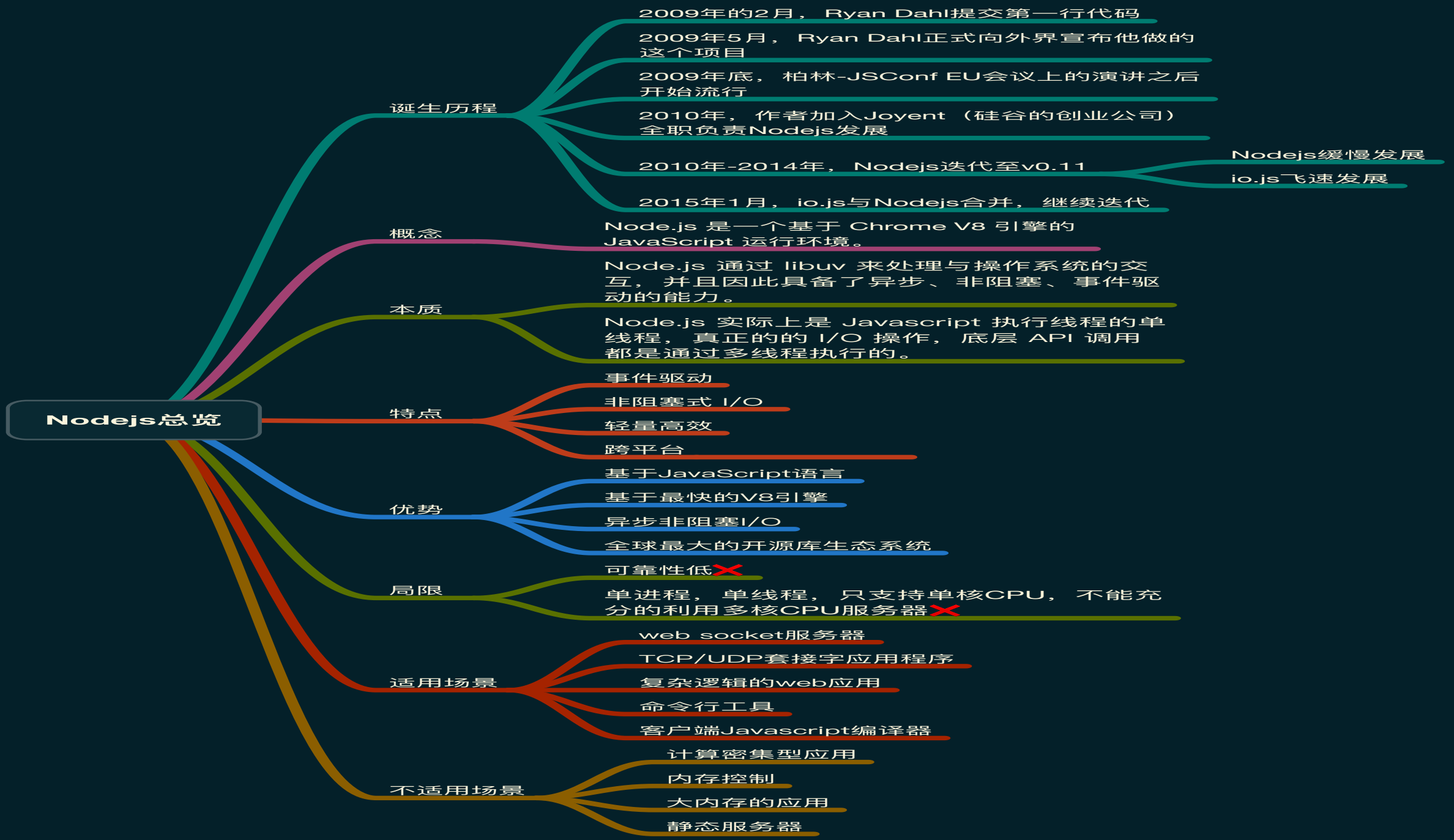
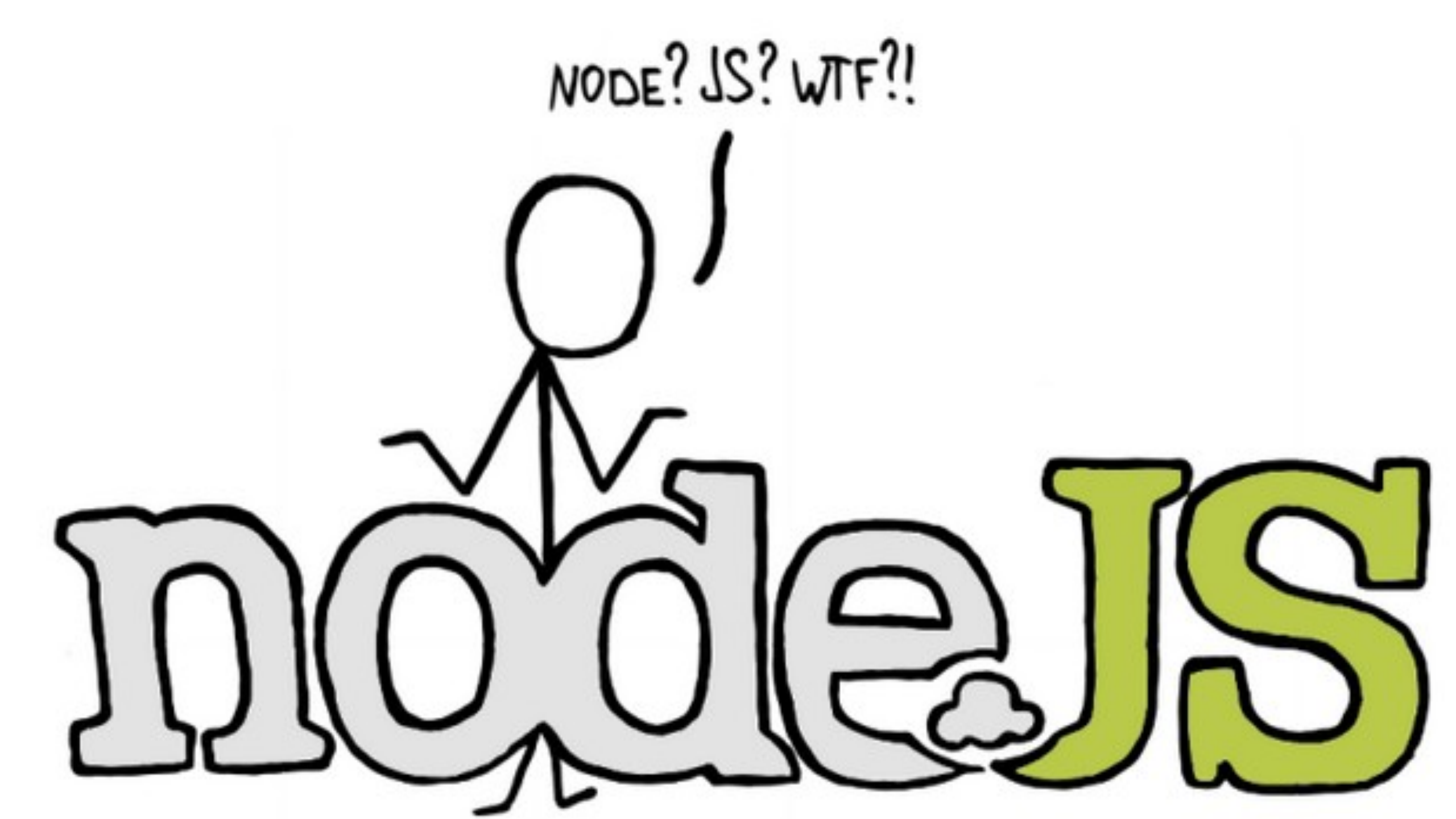
博为峰网校
atstudy.com

Node.js直播课

讲师：黄豪伟


Nodejs总览
诞生历程
2009年的2月， Ryan Dahl提交第一行代码
2009年5月， Ryan Dahl正式向外界宣布他做的这个项目
2009年底， 柏林-JSConf EU会议上的演讲之后开始流行
2010年， 作者加入Joyent （硅谷的创业公司）全职负责Nodejs发展
2010年-2014年， Nodejs迭代至v0.11
Nodejs缓慢发展
io.js飞速发展
2015年1月， io.js与Nodejs合并， 继续迭代
概念
Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。
本质
Node.js 通过 libuv 来处理与操作系统的交互， 并且因此具备了异步、 非阻塞、 事件驱动的能力。
Node.js 实际上是 Javascript 执行线程的单线程， 真正的的 I/O 操作， 底层 API 调用都是通过多线程执行的。
特点
事件驱动
非阻塞式 I/O
轻量高效
跨平台
优势
基于JavaScript语言
基于最快的V8引擎
异步非阻塞I/O
全球最大的开源库生态系统
局限
可靠性低
单进程， 单线程， 只支持单核CPU， 不能充分的利用多核CPU服务器
适用场景
web socket服务器
TCP/UDP套接字应用程序
复杂逻辑的web应用
命令行工具
客户端Javascript编译器
不适用场景
计算密集型应用
内存控制
大内存的应用
静态服务器


NODE? JS? WTF?!
node.JS


Nodejs总览
诞生历程
2009年的2月， Ryan Dahl提交第一行代码
2009年5月， Ryan Dahl正式向外界宣布他做的这个项目
2009年底， 柏林-JSConf EU会议上的演讲之后开始流行
2010年， 作者加入Joyent （硅谷的创业公司）全职负责Nodejs发展
2010年-2014年， Nodejs迭代至v0.11
Nodejs缓慢发展
io.js飞速发展
2015年1月， io.js与Nodejs合并， 继续迭代
概念
Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。
特点
事件驱动
非阻塞式 I/O
轻量又高效
跨平台
优势
基于JavaScript语言
基于最快的V8引擎
异步非阻塞I/O
全球最大的开源库生态系统

Why node.js?

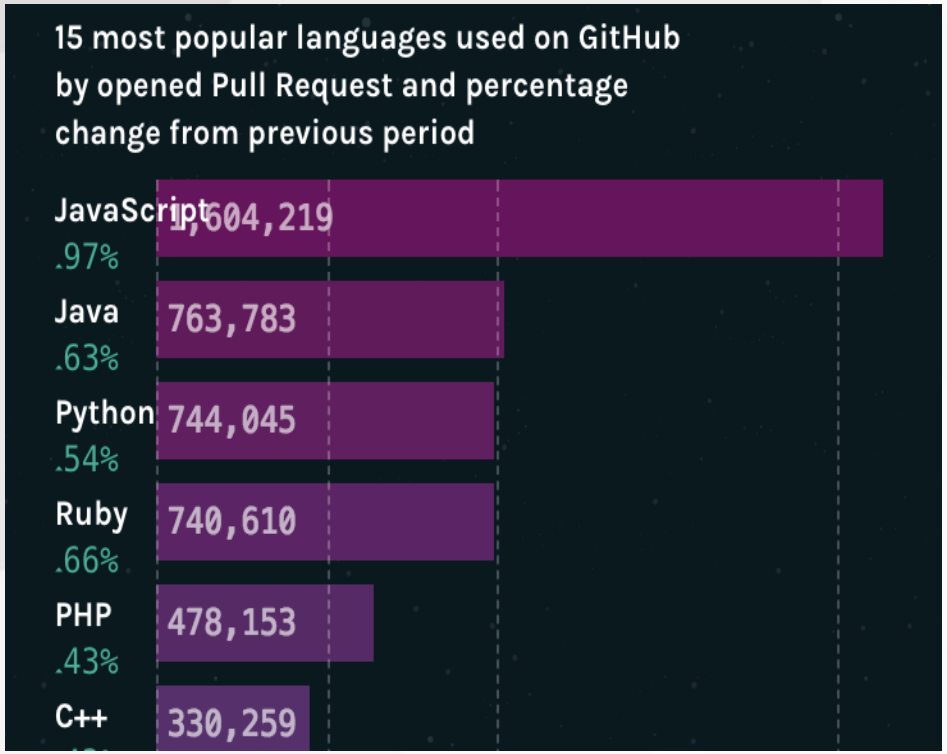
- **Ryan Dahl** (creator of node.js):
  - "I am not happy with the way web servers and apps work today" (apache, php, rails, IIS, etc).
  - "We need something faster, highly scalable".
- Check "**History of node**"
  - http://www.youtube.com/watch?v=SAc0vQCC6UQ

- Thanks Ryan!

- 概述

1. Node.js 是一个基于 **Chrome V8 引擎**的 JavaScript **运行环境**。
2. Node.js 使用了一个**事件驱动、非阻塞式 I/O 的模型**，使其轻量又高效。
3. Node.js 的包**管理器 npm**，是**全球最大**的开源库生态系统。

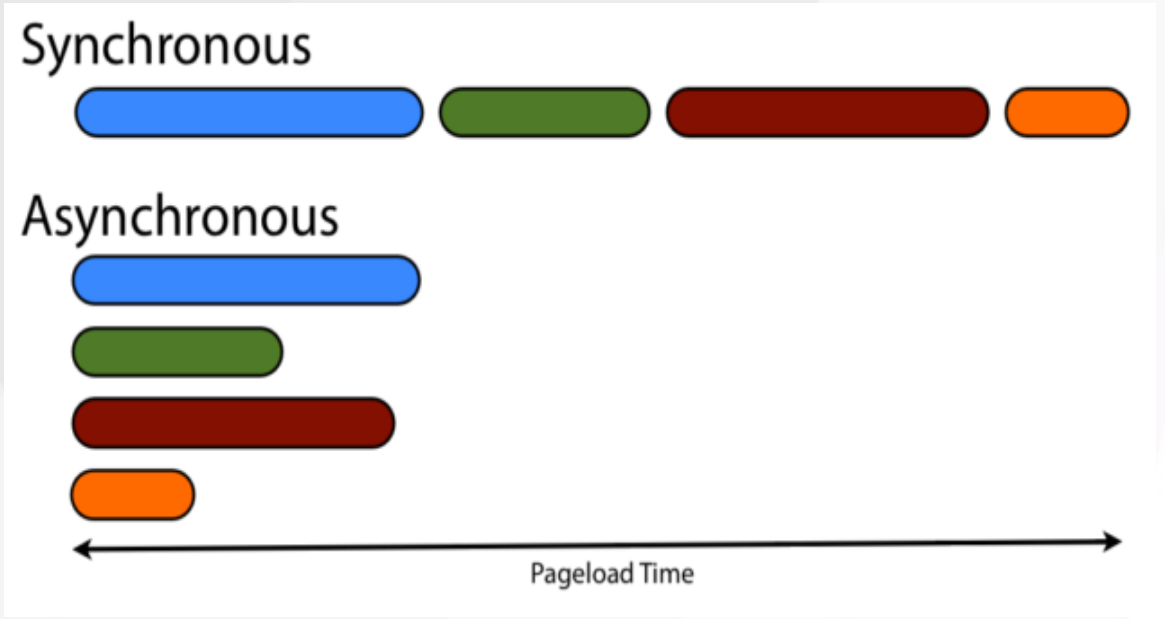
优势



基于最快的V8引擎

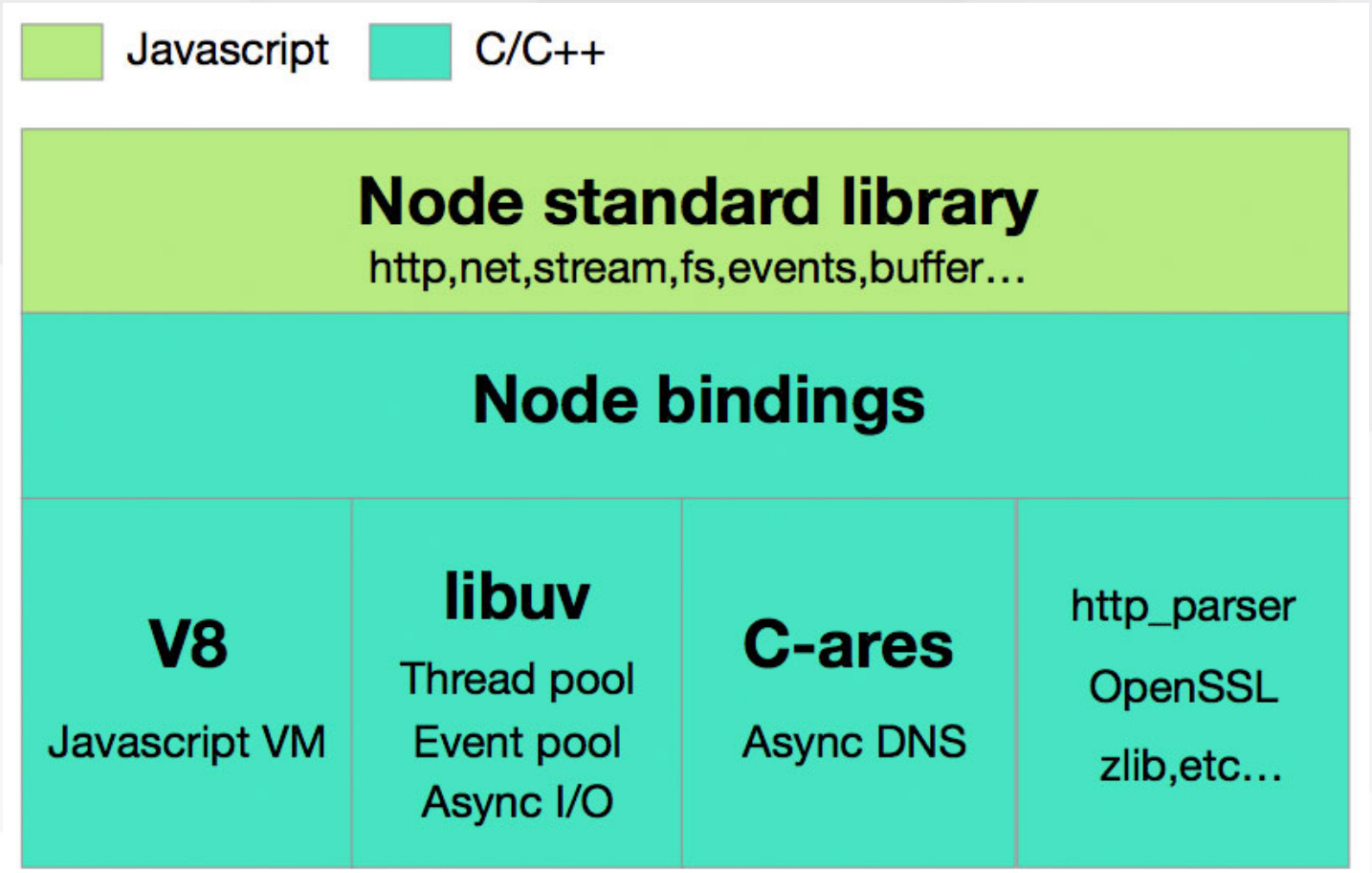
基于最流行的语言JavaScript

优势



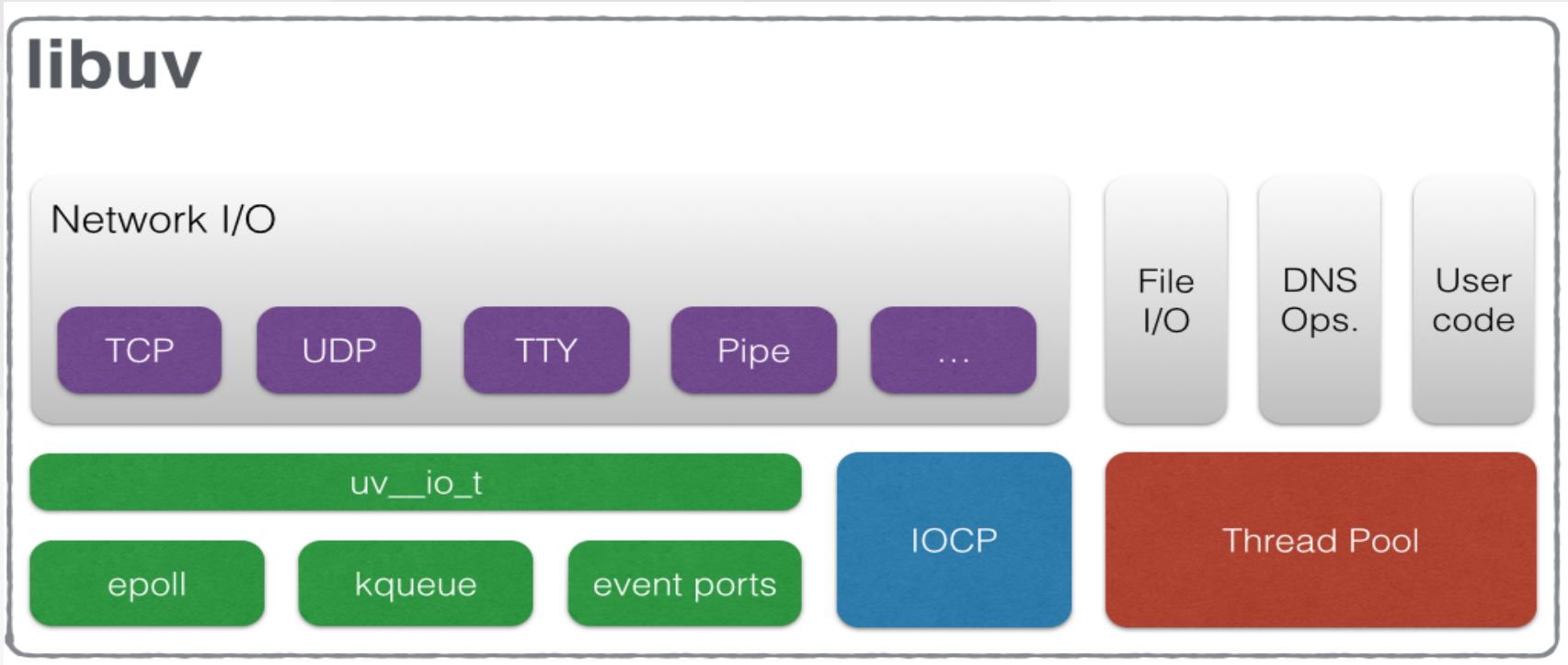
- 全球最大的开源库生态系统
- 异步非阻塞I/O

Nodejs本质



Node.js 的结构。

Node.js 通过 **libuv** 来处理与操作系统的交互，并且因此具备了异步、非阻塞、事件驱动的能力。



Libuv结构

- Node.js 实际上是 Javascript 执行线程的**单线程**，**真正的**的 I/O 操作，底层 API 调用都是通过**多线程**执行的。

Nodejs特点

- 事件驱动
- 非阻塞式 I/O
- 轻量高效
- 跨平台

应用场景


Walmart
ebay
PayPal
DOW JONES
intuit
NETFLIX
LinkedIn
The New York Times
Microsoft
UBER
YAHOO!
Kingfisher

适用场景

1. web socket服务器
2. TCP/UDP套接字应用程序
3. 复杂逻辑的web应用
4. 命令行工具
5. 客户端Javascript编译器

不适用场景

1. 计算密集型应用
2. 内存控制
3. 大内存的应用
4. 静态服务器

学习路线

THANKS!