

Node.js 基础

前端精进 - 后端方向

Node.js 版本

双数是稳定版，单数非稳定版，我们使用 8 以上版本

Node.js 安装

- Windows

- ✓ 去官网下载安装包（不要绿色版）
- ✓ 一路下一步，默认配置即可
- ✓ 安装为了注销账户（可选）
- ✓ 确认命令行里 `node -v` 不出错即可
- ✓ 不方便安装多版本，可以考虑子系统

- Mac

- ✓ 安装 homebrew
- ✓ `brew install node@8` 或者 10
- ✓ 确认 `node -v` 不出错即可
- ✓ 如果你需要多版本，可以使用 `n` 或 `nvm` 两种工具

周边工具

- 推荐安装如下工具

- ✓ nrm 用于切换下载源
- ✓ yarn 和 yrm
- ✓ WebStorm 或者 VSCode
- ✓ ts-node 可以运行 TypeScript 的 node

- Windows

- ✓ Notepad++ 临时打开文件，比 VSCode 快
- ✓ cmdr 可以作为 Git Bash 的代替品

- macOS

- ✓ iTerm2.app
- ✓ oh my zsh

Node.js 不是什么

- 不是 web 框架

- ✓ Node.js 并不是 web 后端框架
- ✓ 所以你不能把 Node.js 与 Flask 或 Spring 对比

- 不是编程语言语言

- ✓ Node.js 并不是后端的 JS
- ✓ 所以你不能把 Node.js 与 Python 或 PHP 对比

Node.js 是什么

- 是一个平台

- ✓ 它将多种技术组合起来
- ✓ 让 JavaScript 也能调用系统接口、开发后端应用

- Node.js 用到了哪些技术

- ✓ V8 引擎
- ✓ libuv
- ✓ C/C++ 实现的 c-ares、http-parser、OpenSSL、zlib 等库

Node.js 技术架构

Node.js API http 模块、fs 模块、stream 模块等				
Node.js bindings 让 JS 和 C/C++ 通信			C/C++ 插件 自定义其他能力	
JS 引擎 V8	跨平台的异步 I/O 能力 libuv	DNS 解析 c-ares	加密解密 OpenSSL	其他... http_parser、zlib 等

随着 Node.js 的版本已经从 0.8 升级到 12.11.1，其架构也在一直变化中
如果你要看源代码，推荐看 [0.10 版本](#)
因为这一版使用了很久一段时间，而且源代码比最新版少很多

如果你想了解更多，可以看 [yjhjstz/deep-into-node](#)

什么是 bindings

- 背景

- ✓ C/C++ 实现了一个 http_parser 库，很高效
- ✓ 你只会写 JS，但是你想调用这个库
- ✓ 直接调用肯定是不能成功的，你需要一个中间的桥梁

- bindings

- ✓ Node.js 用 C++ 对 http_parser 进行封装，使它符合某些要求，封装的文件叫做 http_parser_bindings.cpp
- ✓ 用 Node.js 提供的编译工具将其编译为 .node 文件*
- ✓ JS 代码可以直接 require 这个 .node 文件
- ✓ 这样 JS 就能调用 C++ 库，中间的桥梁就是 binding
- ✓ 由于 Node.js 提供了很多 binding，所以加个 s
- ✓ 这就是 bindings

* 编译成 .node 文件不是必须的，可以是其他的任何可行方式

JS 与 C++ 交互*

- JS 调用 C++ 代码

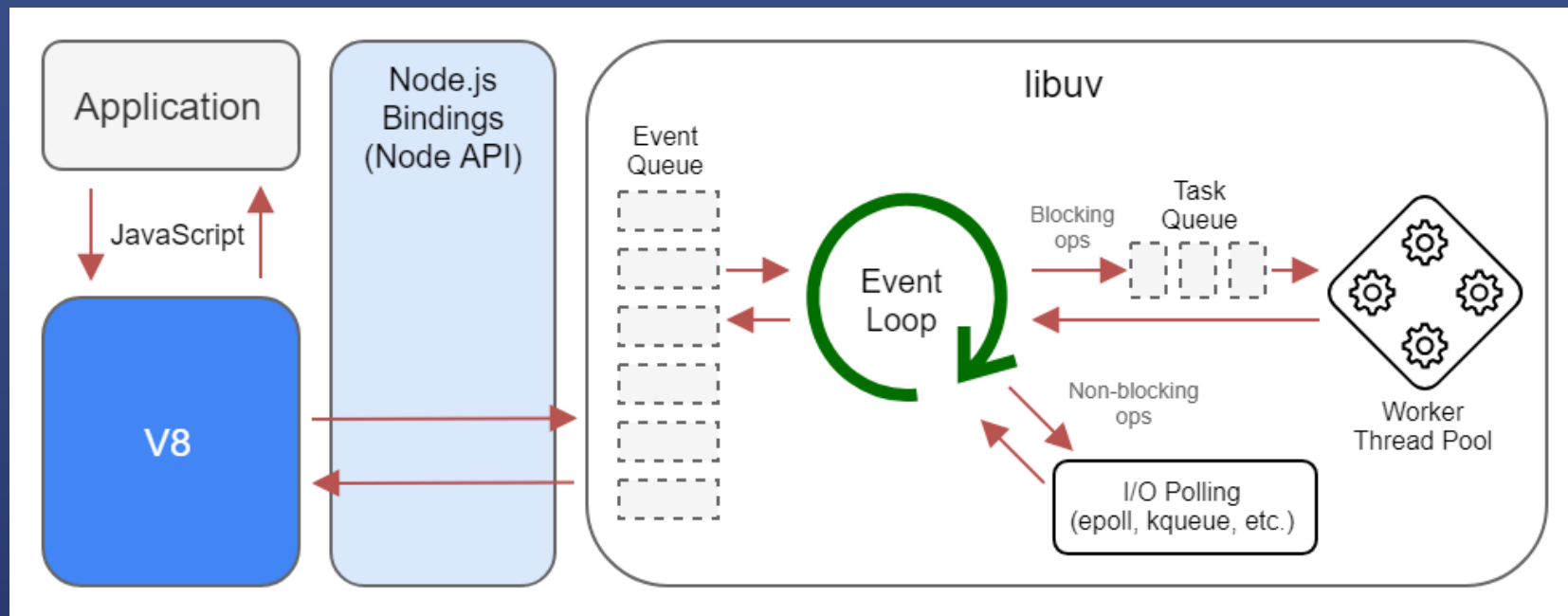
- ✓ 官方示例

- C++ 调用 JS 回调


- ✓ 官方示例




*Node.js 还提供了其他交互方式，我这里只举一例








Node.js 工作流程





Node.js v0.10 的 deps 目录


 nodejs / node

 Watch 2,926  Star 64,663  Fork 15,007








 Code  Issues 809  Pull requests 285  Actions  Projects 7  Security  Insights

Branch: v0.10.0-release node / deps / Create new file Upload files Find file History

This branch is 20146 commits behind master.  Pull request  Compare

 isaacs uv: Upgrade to 5462dab Latest commit 21a9966 on 10 Mar 2013

..

 cares	deps: upgrade cares to 213f2b7	7 years ago
 http_parser	deps: upgrade http_parser to ad3b631	7 years ago
 npm	npm: Upgrade to 1.2.14 (fixed)	7 years ago
 openssl	win/openssl: mark assembled object files as seh safe	7 years ago
 uv	uv: Upgrade to 5462dab	7 years ago
 v8	V8: Reapply floating patches	7 years ago
 zlib	zlib: compile without -ansi	8 years ago

libuv 是什么

- 背景

- ✓ FreeBSD 系统上有 kqueue
- ✓ Linux 系统上有 epoll
- ✓ Windows 系统上有 IOCP
- ✓ Ryan 为了一个跨平台的异步 I/O 库，开始写 libuv
- ✓ libuv 会根据系统自动选择合适的方案

- 功能

- ✓ 可以用于 TCP / UDP / DNS / 文件等的异步操作

V8 是什么

- 功能

- ✓ 将JS源代码变成本地代码并执行
- ✓ 维护调用栈，确保 JS 函数的执行顺序
- ✓ 内存管理，为所有对象分配内存
- ✓ 垃圾回收，重复利用无用的内存
- ✓ 实现 JS 的标准库

- 注意

- ✓ V8 不提供 DOM API
- ✓ V8 执行 JS 是单线程的
- ✓ 可以开启两个线程分别执行 JS
- ✓ V8 本身是包含多个线程的，如垃圾回收为单独线程
- ✓ 自带 event loop 但 Node.js 基于 libuv 自己做了一个

Event Loop 是什么

- 什么是 Event

- ✓ 计时器到期了
- ✓ 文件可以读取了、读取出错了
- ✓ socket 有内容了、关闭了

- 什么是 Loop

- ✓ loop 就是循环，比如 while(true) 循环
- ✓ 由于事件是分优先级的，所以处理起来也是分先后的
- ✓ 所以 Node.js 需要按顺序轮询每种事件
- ✓ 这种轮询往往都是循环的，1->2->3->1->2->3

- Event Loop

- ✓ 操作系统可以触发事件，JS 可以处理事件
- ✓ Event Loop 就是对事件处理顺序的管理

举例

- 三种不同的事件

```
setTimeout(f1, 100)
```

```
fs.readFile('/1.txt', f2)
```

```
server.on('close', f3)
```

- 如果同时触发，Node 会怎么办

- ✓ 肯定会有某种顺序（优先级）
- ✓ 这种顺序应该是认为规定的

Event Loop

- 顺序示意图

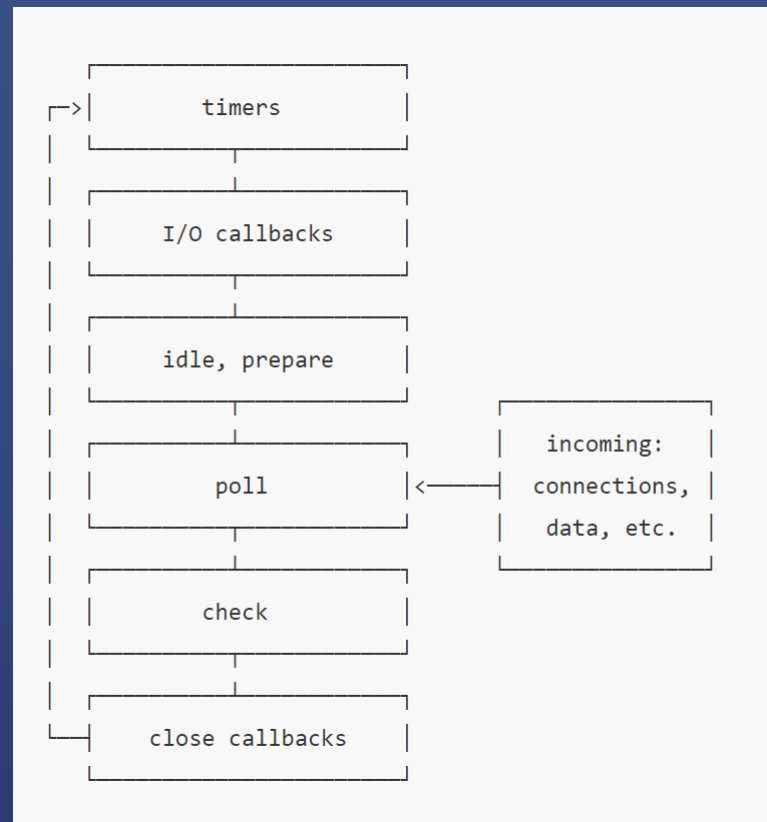
- ✓ 我翻译过[官方文档](#)
- ✓ 读官方文档是最好的方式

- 重点阶段

- ✓ timers 检查计时器
- ✓ poll 轮询，检查系统事件
- ✓ check 检查 setImmediate 回调
- ✓ 其他阶段用得较少

- 注意

- ✓ 大部分时间，Node.js 都停在 poll 轮询阶段
- ✓ 大部分事件都在 poll 阶段被处理，如文件、网络请求



总结

- 用 libuv 进行异步 I/O 操作
- 用 event loop 管理事件处理顺序
- 用 C/C++ 库高效处理 DNS/HTTP...
- 用 bindings 让 JS 能和 C/C++ 沟通
- 用 V8 运行 JS
- 用 Node.js 标准库简化 JS 代码
- 这就是 Node.js

回顾 Node.js 技术架构

Node.js API				
http 模块、fs 模块、stream 模块等				
Node.js bindings			C/C++ 插件	
让 JS 和 C/C++ 通信			自定义其他能力	
JS 引擎 V8	跨平台的异步I/O能力 libuv	DNS 解析 c-ares	加密解密 OpenSSL	其他... http_parser、zlib等

我们大部分时候，只需要学习 Node.js 标准库即可
对于其他模块，有一个大概的了解就行
等你对 Node.js 内部实现感兴趣的时候，再去了解标准库之外的东西
不过你可能要写学好 C/C++

Node.js API

官方提供给我们用的函数

API 文档

- 官方地址

- ✓ 英文文档 nodejs.org/api/
- ✓ 中文文档 nodejs.cn/api/

- 民间版本

- ✓ devdocs.io
- ✓ 进入之后开启 Node.js 10 LTS
- ✓ 搜索功能非常方便
- ✓ 可开启暗黑主题
- ✓ 可离线观看

API 到底有哪些功能

Assertion

Testing

Async Hooks

Buffer

Child Processes

Cluster

Console

Crypto

Debugger

DNS

Errors

Events

File System

Globals

HTTP

HTTP/2

HTTPS

Inspector

i18n

Net

OS

Path

Performance Hooks

Process

Query Strings

Readline

REPL

Report

Stream

String Decoder

Timers

TLS/SSL

Trace Events

TTY

UDP/Datagram

URL

Utilities

V8

VM

Worker Threads*

Zlib

学习路线

- 基础 - Web - 框架

- ✓ 先学基础，以任务为导向学习
- ✓ 逐个学习文件、HTTP、Stream 等模块
- ✓ 再学 Web，学习数据库、AJAX 相关知识
- ✓ 最后学框架，以项目为导向学习
- ✓ 以 Express 为切入点，制作完整的网站

- 三个约定

- ✓ 希望大家记笔记，写博客
- ✓ CRM 学习法贯穿整个学习过程
- ✓ 学习我的调试工具和思路

参考文档

- ✓ [C++ binding with Node.js](#)
- ✓ [Understanding Worker Threads in Node.js](#)