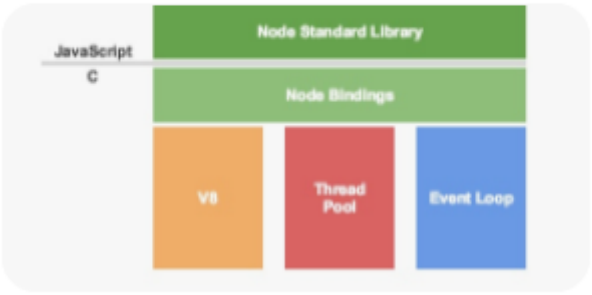
c) 基本原理

下面是一张 Node.js 早期的架构图，来自 Node.js 之父 Ryan Dahl 的演讲稿，在今天依然不过时，它简要的介绍了 Node.js 是基于 Chrome V8引擎构建的，由事件循环（Event Loop）分发 I/O 任务，最终工作线程（Work Thread）将任务丢到线程池（Thread Pool）里去执行，而事件循环只要等待执行结果就可以了。

1 人喜欢

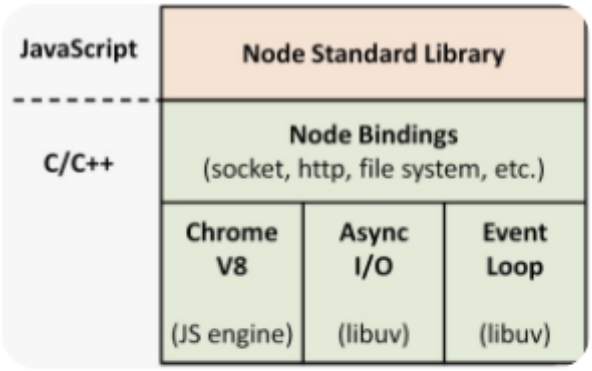


核心概念

- Chrome V8 是 Google 发布的开源 JavaScript 引擎，采用 C/C++ 编写，在 Google 的 `Chrome` 浏览器中被使用。Chrome V8 引擎可以独立运行，也可以用来嵌入到 C/C++ 应用程序中执行。
- Event Loop 事件循环（由 `libuv` 提供）
- Thread Pool 线程池（由 `libuv` 提供）

梳理一下

- Chrome V8 是 JavaScript 引擎
- Node.js 内置 Chrome V8 引擎，所以它使用的 JavaScript 语法
- JavaScript 语言的一大特点就是单线程，也就是说，同一个时间只能做一件事
- 单线程就意味着，所有任务需要排队，前一个任务结束，才会执行后一个任务。如果前一个任务耗时很长，后一个任务就不得不一直等着。
- 如果排队是因为计算量大，CPU 忙不过来，倒也算了，但是很多时候 CPU 是闲着的，因为 I/O 很慢，不得不等着结果出来，再往下执行
- CPU 完全可以不管 I/O 设备，挂起处于等待中的任务，先运行排在后面的任务
- 将等待中的 I/O 任务放到 Event Loop 里
- 由 Event Loop 将 I/O 任务放到线程池里
- 只要有资源，就尽力执行



5 人喜欢

核心

- Chrome V8 解释并执行 JavaScript 代码（这就是为什么浏览器能执行 JavaScript 原因）
- `libuv` 由事件循环和线程池组成，负责所有 I/O 任务的分发与执行

在解决并发问题上，异步是最好的解决方案，可以拿排队和叫号机来理解

- 排队：在排队的时候，你除了等之外什么都干不了
- 叫号机：你要做的是先取号码，等轮到你的时候，系统会通知你，这中间，你可以做任何你想做的事儿

Node.js 其实就是帮我们构建类似的机制。我们在写代码的时候，实际上就是取号的过程，由 Event Loop 来接受处理，而真正执行操作的是具体的线程池里的 I/O 任务。之所以说 Node.js 是单线程，就是因为在接受任务的时候是单线程的，它无需进程/线程切换上下文的成本，非常高效，但它在执行具体任务的时候是多线程的。