---

link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=30 sentences=21, words=108

---

1. Node能够解决什么问题？#

- Node的首要目标是提供一种简单的，用于创建高性能服务器的开发工具
- Web服务器的瓶颈在于并发的用户量，对比Java和Php的实现方式

2. Node是什么？#

- Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境,让JavaScript的执行效率与低端的C语言的相近的执行效率。。
- 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。
- Node.js 的包管理器 npm，是全球最大的开源库生态系统。

3. Node特点

3.1 为什么JavaScript是单线程？

- 这是由 Javascript 这门脚本语言的用途决定的。
- Web Worker并没有改变 JavaScript 单线程的本质。

3.2 浏览器模型

- 用户界面-包括地址栏、前进/后退按钮、书签菜单等
- 浏览器引擎-在用户界面和呈现引擎之间传送指令
- 呈现引擎-又称渲染引擎，也被称为浏览器内核，在线程方面又称为UI线程
- 网络-用于网络调用，比如 HTTP 请求
- 用户界面后端-用于绘制基本的窗口小部件,UI线程和JS共用一个线程
- JavaScript解释器-用于解析和执行 JavaScript 代码
- 数据存储-这是持久层。浏览器需要在硬盘上保存各种数据，例如 Cookie

3.3 除JS线程和UI线程之外的其它线程

- 浏览器事件触发线程
- 定时触发器线程
- 异步HTTP请求线程

3.4 任务队列

1. 所有同步任务都在主线程上执行，形成一个执行栈
2. 主线程之外，还存在一个任务队列。只要异步任务有了运行结果，就在任务队列之中放置一个事件。
3. 一旦执行栈中的所有 **同步任务** 执行完毕，系统就会读取 **任务队列**，看看里面有哪些事件。那些对应的异步任务，于是结束等待状态，进入执行栈，开始执行。
4. 主线程不断重复上面的第三步。

3.5. Event Loop

主线程从 **任务队列** 中读取事件，这个过程是循环不断的，所以整个的这种运行机制又称为Event Loop(事件循环)

3.6. Node.js的Event Loop

1. V8引擎解析JavaScript脚本。
2. 解析后的代码，调用Node API。
3. libuv库负责Node API的执行。它将不同的任务分配给不同的线程，形成一个Event Loop（事件循环），以异步的方式将任务的执行结果返回给V8引擎。
4. V8引擎再将结果返回给用户。

3.7. 同步与异步

同步和异步关注的是消息通知机制

- 同步就是发出调用后，没有得到结果之前，该调用不返回，一旦调用返回，就得到返回值了。 简而言之就是调用者主动等待这个调用的结果
- 而异步则相反，调用者在发出调用后这个调用就直接返回了，所以没有返回结果。换句话说当一个异步过程调用发出后，调用者不会立刻得到结果，而是调用发出后，被调用者通过状态、通知或回调函数处理这个调用。

3.8. 阻塞与非阻塞

阻塞和非阻塞关注的是程序在等待调用结果（消息，返回值）时的状态.

- 阻塞调用是指调用结果返回之前，当前线程会被挂起。调用线程只有在得到结果之后才会返回。
- 非阻塞调用指在不能立刻得到结果之前，该调用不会阻塞当前线程。

3.9. 组合

同步异步取决于被调用者，阻塞非阻塞取决于调用者

- 同步阻塞
- 异步阻塞
- 同步非阻塞
- 异步非阻塞

4.什么场合下应该考虑使用Node框架

当应用程序需要处理大量并发的输入输出，而在向客户端响应之前，应用程序并不需要进行非常复杂的处理。

- 聊天服务器
- 电子商务网站