Node.js到底是用来做什么的

1. Node.js到底是什么

Node.js就是JavaScript的编译环境，它存在的目的就是为了让JavaScript可以和其他的后端语言一样能够在浏览器上运行,可以让前端语言JavaScript在写完之后交给Node.js进行编译和解释，它的存在对于JavaScript有了质的飞跃

2. V8引擎

我们都知道计算机处理器智能识别机器语言，而JavaScript是一门高级语言，计算机并不能直接读懂。所以我们需要所谓的引擎来将其转化成计算机所能理解的语言。举例说明，JavaScript本身并没有read这么一个function。然而通过v8，我们可以将其绑定到一个用c++写的read callback上，从而通过JavaScript我们也可以直接加载文件了。

于是，借助于v8种种便利的功能，Node.js诞生了。

3. 数据的请求和处理

首先我们要注意的是**浏览器给网站发请求的过程一直没怎么变过**。当浏览器给网站发了请求。服务器收到了请求，然后开始搜寻被请求的资源。如果有需要，服务器还会查询一下数据库，最后把响应结果传回浏览器。不过，在传统的web服务器中，每一个请求都会让服务器创建一个新的进程来处理这个请求。后来有了Ajax，我们就不用每次都请求一个完整的新页面了，取而代之的是，每次只请求需要的部分页面信息就可以了。这显然是一个进步。但是比如你要建一个类似微博的社交网站，导致的结果是你的好友会随时的推送新的状态，然后你的新鲜事会实时自动刷新。要达成这个需求，我们需要让用户一直与服务器保持一个有效连接。目前最简单的实现方法，就是让用户和服务器之间保持长轮训（long polling）。

4. 使用Node.js的优劣（很重要）

优势

- 采用事件驱动、异步编程，为网络服务而设计。其实Javascript的匿名函数和闭包特性非常适合事件驱动、异步编程。而且JavaScript也简单易学，很多前端设计人员可以很快上手做后端设计。
- Node.js非阻塞模式的IO处理给Node.js带来在相对低系统资源耗用下的高性能与出众的负载能力，非常适合用作依赖其它IO资源的中间层服务。
- • Node.js轻量高效，可以认为是数据密集型分布式部署环境下的实时应用系统的完美解决方案。Node非常适合如下情况：在响应客户端之前，您预计可能有很高的流量，但所需的服务器端逻辑和

处理不一定很多。

缺点

- 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。一旦这个进程崩掉，那么整个web服务就崩掉了。
- 开启多个进程，每个进程绑定不同的端口，用反向代理服务器如Nginx 做负载均衡，好处是我们可以借助强大的 Nginx 做一些过滤检查之类的操作，同时能够实现比较好的均衡策略，• 但坏处也是显而易见——我们引入了一个间接层。
- 多进程绑定在同一个端口侦听。在Node.js中，提供了进程间发送“文件句柄” 的功能。一个进程负责监听、接收连接，然后把接收到的连接平均发送到子进程中去处理。

5. 适用场景

SON APIs——构建一个Rest/JSON API服务，Node.js可以充分发挥其非阻塞IO模型以及JavaScript对JSON的功能支持(如JSON.stringfy函数)单页面、多Ajax请求应用——如Gmail，前端有大量的异步请求，需要服务后端有极高的响应速度基于Node.js开发Unix命令行工具——Node.js可以大量生产子进程，并以流的方式输出，这使得它非常适合做Unix命令行工具流式数据——传统的Web应用，通常会将HTTP请求和响应看成是原子事件。而Node.js会充分利用流式数据这个特点，构建非常酷的应用。如实时文件上传系统transloadit准实时应用系统——如聊天系统、微博系统，但Javascript是有垃圾回收机制的，这就意味着，系统的响应时间是不平滑的(GC垃圾回收会导致系统这一时刻停止工作)。如果想要构建硬实时应用系统，Erlang是个不错的选择。