作者：贾厂长  
链接：https://www.zhihu.com/question/33578075/answer/56951771  
来源：知乎  
著作权归作者所有，转载请联系作者获得授权。

如果你去年注意过技术方面的新闻，我敢说你至少看到node.js不下一两次。那么问题来了“node.js是什么？”。有些人没准会告诉你“这是一种通过JavaScript语言开发web服务端的东西”。如果这种晦涩解释还没把你搞晕，你没准会接着问：“为什么我们要用node.js？”，别人一般会告诉你：node.js有非阻塞，事件驱动I/O等特性，从而让高并发（high concurrency）在的轮询（Polling）和comet构建的应用中成为可能。

当你看完这些解释觉得跟看天书一样的时候，你估计也懒得继续问了。不过没事。我这篇文章就是在避开高端术语的同时，帮助你你理解node.js的。

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（比如Apache），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。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FriendFeed这样的社交网站（类似人人网那样的刷朋友新鲜事的网站），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询（long polling）。

HTTP请求不是持续的连接，你请求一次，服务器响应一次，然后就完了。长轮训是一种利用HTTP模拟持续连接的技巧。具体来说，只要页面载入了，不管你需不需要服务器给你响应信息，你都会给服务器发一个Ajax请求。这个请求不同于一般的Ajax请求，服务器不会直接给你返回信息，而是它要等着，直到服务器觉得该给你发信息了，它才会响应。比如，你的好友发了一条新鲜事，服务器就会把这个新鲜事当做响应发给你的浏览器，然后你的浏览器就刷新页面了。浏览器收到响应刷新完之后，再发送一条新的请求给服务器，这个请求依然不会立即被响应。于是就开始重复以上步骤。利用这个方法，可以让浏览器始终保持等待响应的状态。虽然以上过程依然只有非持续的Http参与，但是我们模拟出了一个看似持续的连接状态

我们再看传统的服务器（比如Apache）。每次一个新用户连到你的网站上，你的服务器就得开一个连接。每个连接都需要占一个进程，这些进程大部分时间都是闲着的（比如等着你好友发新鲜事，等好友发完才给用户响应信息。或者等着数据库返回查询结果什么的）。虽然这些进程闲着，但是照样占用内存。这意味着，如果用户连接数的增长到一定规模，你服务器没准就要耗光内存直接瘫了。

这种情况怎么解决？解决方法就是刚才上边说的：**非阻塞**和**事件驱动**。这些概念在我们谈的这个情景里面其实没那么难理解。你把非阻塞的服务器想象成一个loop循环，这个loop会一直跑下去。一个新请求来了，这个loop就接了这个请求，把这个请求传给其他的进程（比如传给一个搞数据库查询的进程），然后响应一个回调（callback）。完事了这loop就接着跑，接其他的请求。这样下来。服务器就不会像之前那样傻等着数据库返回结果了。

如果数据库把结果返回来了，loop就把结果传回用户的浏览器，接着继续跑。在这种方式下，你的服务器的进程就不会闲着等着。从而在理论上说，同一时刻的数据库查询数量，以及用户的请求数量就没有限制了。服务器只在用户那边有事件发生的时候才响应，这就是**事件驱动。**

FriendFeed是用基于Python的**非阻塞**框架Tornado (知乎也用了这个框架) 来实现上面说的新鲜事功能的。不过，Node.js就比前者更妙了。Node.js的应用是通过javascript开发的，然后直接在Google的变态V8引擎上跑。用了Node.js，你就不用担心用户端的请求会在服务器里跑了一段能够造成阻塞的代码了。因为javascript本身就是事件驱动的脚本语言。你回想一下，在给前端写javascript的时候，更多时候你都是在搞事件处理和回调函数。javascript本身就是给事件处理量身定制的语言。

Node.js还是处于初期阶段。如果你想开发一个基于Node.js的应用，你应该会需要写一些很底层代码。但是下一代浏览器很快就要采用WebSocket技术了，从而长轮询也会消失。在Web开发里，Node.js这种类型的技术只会变得越来越重要。