让我们现在来谈谈一项最近得到了理所应当的大肆宣传的技术—Node.js。现在Node.js是硅谷最热门的新技术，微软、VMWare、易趣、雅虎等很多顶尖的技术公司都普遍在使用这项技术，对软件开发技术人员来说这是一项非常完美的技术，能够给他们带来意想不到的就业机会。

你可能听说过Node.js，但是只知道它与实时或者高扩展性的应用程序有关，你也可能听说过Node.js是针对服务器端的JavaScript脚本语言（并且你很想知道为什么所有人需要它！），或者你知道Node.js是什么，但是不确定什么时候或者为什么要去使用它。慢慢来，我将会解释这所有问题。

让我从头开始来说。

**网站是不断变化的**

网络曾经是与消费有关的，像浏览网页，观看视频，欣赏猫的图片，当然，现在它仍然与猫的图片有关……但是现在网络与交流有更大的联系。世界各地的用户希望彼此能够交流，并且能实时地交流。聊天、游戏、社交媒介的不断更新、团队协作所有这些特征都要求用户、客户端和服务器端能够通过网络实时交流。而且，实时交流必须是大规模的，能够支持成百、上千甚至数百万的用户。

所以，软件工程师应该做什么来实现这些呢？我们需要客户端和服务器端能实时交流，那意味着我们需要快速的持续稳定的输入/输出。任何有网站开发经验的人都知道HTTP(超文本传送协议)在创建时是没有这个作用的。大量的客户端不断地同时访问一个服务器是非常慢并且效率低下的。为了保证大规模的实时交流，服务器必须能够推送数据给客户端而不是繁重的HTTP请求/响应模式，并且我们必须保证这些轻松推送的交流工作从软件发展立足点上来说在某种程度上是大规模的，可持续的和可使用的。

**Node.js入门**

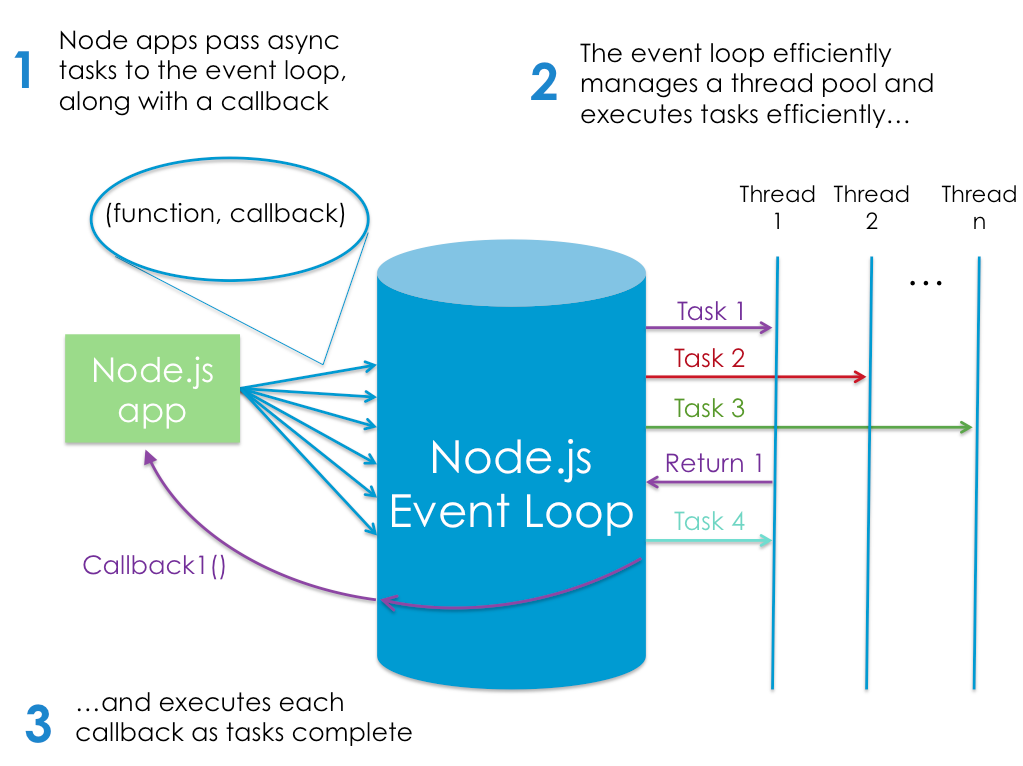
Node.js 是一个事件驱动，服务器端的JavaScript环境。Node结点用V8引擎运行JavaScript,V8引擎是谷歌公司为他们的谷歌浏览器开发的。借助于V8，Node得以提供一个服务器端的运行环境来快速的编译和执行JavaScript语言。它速度的提升主要是因为V8把JavaScript编译为本地机器代码（直接解释执行），而不是编绎为字节码或中间代码（再解释执行）。Node.js是开源的，跨平台的，能够运行在Mac OSX，Windows和Linux等操作系统上。

(OurJS注\* 关于NodeJS性能方面的评测，可参见： [PayPal为什么从Java迁移到Node.js](http://ourjs.com/detail/52a914f0127c763203000008)[, Ngix\_Lua, Node.JS Python三者性能相当，均比php快近一倍](http://ourjs.com/detail/52954f16f45056c314000001), [性能测评：Node.JS比Java EE快20%](http://ourjs.com/detail/52726d92031db63703000001))

但是，JavaScript运行在服务器端？这是为什么呢？虽然JavaScript历来降级在web浏览器中做低贱的工作，它实际上是一个功能完整的编程语言,能够比传统的c++等语言做更多的事。Ruby,或者Java. 此外，JavaScript有出色的事件模式优势，理想的异步编程。JavaScript也是一个无处不在的语言，成千上万的开发者都知道它。这降低了node . js的学习曲线，因为大多数开发者不需要学习新的语言去开发Node.js应用程序。

**异步编程的简单方法**

除了闪电般快速的JavaScript执行，Node.js背后的真正魔力是个叫事件循环的机制。为了能使大规模的客户使用流畅，Node.js中所有的I/O集中操作都是异步的。传统的线程方法来达到异步编程是难以处理的，也为大量的客户端产生不必要的的内存占用（一个客户端产生一个线程，内存占用就相应地增加）。为了避免这种低效率并且我们都知道线程编程的困难，Node.js就包含了一个事件循环机制(Event Loop)，为你管理所有的异步操作。当一个Node应用程序需要去完成一个块操作（I/O输入输出操作，大量的计算等等），它就传送一个异步任务给事件循环，伴随着一个回调函数，然后继续执行程序的余下部分。事件循环跟踪着异步操作，当完成时执行异步操作所传送的的回调函数，返回它的结果的给应用程序。这种机制允许你管理大量的操作，比如客户端的连接或者计算，让事件循环高效的管理线程池并且优化任务执行的速度。当然，把这个责任留给事件循环后，Node.js开发者会特别轻松，然后他们可以专注于实现他们的程序功能。



注;这里是图解讲解Node.js的事件循环生命周期

能够简化异步编程的Node.js对开发者来说是强大的工具 。通过使用Node.js,你可以开发复杂的应用程序链接千百万的客户端，因为应用程序处理客户端请求是仿冒密集型的工作要管理输入 / 输出和事件循环计算。

**社区驱动**

除了它固有的能力，Node.js有一个繁荣的开源社区创造了很多出色的模块给Node.js应用程序增加了许多额外的功能，其中最著名的是socket.io,它是一个管理客户端和服务器之间的持久连接的模块，能够启用服务器推送实时更新到客户端。Socket.io抽象了这个技术使开发人员可以不用理会这些连接，它自动使用最好的技术可用于一个特定的客户端（如果浏览器支持的话就使用网络套接字，如果不支持就使用JSONP或者Ajax长连接）。这个神奇的技术让你用很少的代码容易地编写实时应用程序。例如这个高扩展性,实时聊天服务器：

server = require('http').createServer();  
  
server.listen(8080);  
  
var io = require('socket.io').listen(server);  
  
io.sockets.on('connection', function(socket) {   
  
 // when message received, send it to all connected clients  
  
 socket.on('message', function(message){  
  
 console.log("Received message: " + message +   
  
 " - from client " + socket.id);  
  
 io.sockets.emit('chat', socket.id, message);  
 });  
});

只有短短的十行代码！相应的客户端代码一样简单，稍微想一下这个可能性！ [学习Node.js的例子](https://www.udemy.com/learn-nodejs-by-example/?tc=blog.nodejs.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post7970&utm_campaign=content-marketing-blog&xref=blog&utm_source=blog&utm_medium=udemyads&utm_content=post7970&utm_campaign=content-marketing-blog&xref=blog)，现在开始编写你未来的可伸缩的、实时的应用程序。这种高水平的课程提供详细的演示，教程和深入的代码练习，让你即刻动手编写Node.js应用程序。

关于作者

康斯坦丁亚伦Cois在卡内基梅隆大学领导一个软件工程团队。另外，他是一位兼职教授给研究生教授软件工程课程。亚伦在2007年获得博士学位，为生物医学应用程序开发先进的算法和软件系统。他现在的研究领域是web应用程序，实时应用系统和云计算

博客地址: [http://www.codehenge.net](http://www.codehenge.net/)

邮箱: cacois@gmail.com