# **####** Node 的优点和缺点提出了自己的看法：

（优点）因为Node 是基于事件驱动和无阻塞的，所以非常适合处理并发请求，

因此构建在Node 上的代理服务器相比其他技术实现（如Ruby）的服务器表现要好得多。

此外，与Node 代理服务器交互的客户端代码是由javascript 语言编写的，

因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

（缺点）Node 是一个相对新的开源项目，所以不太稳定，它总是一直在变，

而且缺少足够多的第三方库支持。看起来，就像是Ruby/Rails 当年的样子。

需求：实现一个页面操作不会整页刷新的网站，并且能在浏览器前进、后退

时正确响应。给出你的技术实现方案？

至少给出自己的思路（url-hash,可以使用已有的一些框架history.js 等）

# **#### Node.js 的适用场景？**

1)、实时应用：如在线聊天，实时通知推送等等（如socket.io）

2)、分布式应用：通过高效的并行I/O 使用已有的数据

3)、工具类应用：海量的工具，小到前端压缩部署（如grunt），大到桌面图形界面应用程

序

4)、游戏类应用：游戏领域对实时和并发有很高的要求（如网易的pomelo 框架）

5)、利用稳定接口提升Web 渲染能力

6)、前后端编程语言环境统一：前端开发人员可以非常快速地切入到服务器端的开发（如著 名的纯Javascript 全栈式MEAN 架构）

# **#### 1、什么是错误优先的回调函数？**

错误优先的回调函数用于传递错误和数据。第一个参数始终应该是一个错误对象， 用于检查程序是否发生了错误。其余的参数用于传递数据。

s.readFile(filePath, function(err, data) {

if (err) {

//handle the error

}

// use the data object

});

解析：这个题目的主要作用在于检查被面试者对于Node中异步操作的一些基本知识的掌握。

# **#### 3、如何用Node监听80端口**

这题有陷阱！在类Unix系统中你不应该尝试去监听80端口，因为这需要超级用户权限。 因此不推荐让你的应用直接监听这个端口。

　　目前，如果你一定要让你的应用监听80端口的话，你可以有通过在Node应用的前方再增加一层反向代理 （例如nginx）来实现，如下图所示。否则，建议你直接监听大于1024的端口。

反向代理指的是以代理服务器来接收Internet上的连接请求，然后将请求转发给内部网络上的服务器， 并且将服务器返回的结果发送给客户端。

# #### 4.什么是事件循环，nodejs底层分析

Node采用的是单线程的处理机制（所有的I/O请求都采用非阻塞的工作方式），至少从Node.js开发者的角度是这样的。 而在底层，Node.js借助libuv来作为抽象封装层， 从而屏蔽不同操作系统的差异，Node可以借助livuv来来实现多线程。下图表示了Node和libuv的关系。

Libuv库负责Node API的执行。它将不同的任务分配给不同的线程，形成一个事件循环， 以异步的方式将任务的执行结果返回给V8引擎。可以简单用下面这张图来表示。

每一个I/O都需要一个回调函数——一旦执行完便推到事件循环上用于执行。

　解释：这用于检查Node.js的底层知识，例如什么是libuv，它的作用是什么。

Event Loop: http://www.ruanyifeng.com/blog/2014/10/event-loop.html

# **####** 5、哪些工具可以用来保证一致性的代码风格

你可以选择如下的工具：

JSLint

JSHint

ESLint

JSCS - 推荐

　　在团队开发中，这些工具对于编写代码非常的有帮助，能够帮助团队开发者强制执行规定的风格指南， 还能够通过静态分析捕获常见的错误。

　　解析：用于检查被面试者是否有大型项目开发经验。

# **####** 6、运算错误与程序员错误的区别

运算错误并不是bug，这是和系统相关的问题，例如请求超时或者硬件故障。而程序员错误就是所谓的bug。

解析：这个题目和Node关系并不大，用于考察面试者的基础知识。

# **####** 7、使用NPM有哪些好处？

通过NPM，你可以安装和管理项目的依赖，并且能够指明依赖项的具体版本号。 对于Node应用开发而言，你可以通过package.json文件来管理项目信息，配置脚本， 以及指明项目依赖的具体版本。

解析：它能考察面试者使用npm命令的基础知识和Node.js开发的实际经验。

# **####** 8、什么是Stub？举个使用场景

Stub是用于模拟一个组件或模块的函数或程序。在测试用例中， 简单的说，你可以用Stub去模拟一个方法，从而避免调用真实的方法， 使用Stub你还可以返回虚构的结果。你可以配合断言使用Stub。

举个例子，在一个读取文件的场景中，当你不想读取一个真正的文件时：

var fs = require('fs');

var readFileStub = sinon.stub(fs, 'readFile', function (path, cb) {

return cb(null, 'filecontent');

});

expect(readFileStub).to.be.called;

readFileStub.restore();

在单元测试中：Stub是完全模拟一个外部依赖，而Mock常用来判断测试通过还是失败。

# **####** 8、什么是测试金字塔？

测试金字塔指的是： 当我们在编写测试用例时，底层的单元测试应该远比上层的端到端测试要多。

当我们谈到HTTP API时，我们可能会涉及到：

有很多针对模型的底层单元测试

但你需要测试模型间如何交互时，需要减少集成测试

# **####** 9、你最喜欢的HTTP框架以及原因

这题没有唯一的答案。本题主要考察被面试者对于他所使用的Node框架的理解程度， 考察他是否能够给出选择该框架的理由，优缺点等。常用的HTTP框架你可以参考这个网站。