## Node.js面试题列表

什么是错误优先的回调函数？

如何避免回调地狱？

如何用Node来监听80端口？

什么是事件循环？

哪些工具可以用来保证一致的编程风格？

什么是测试金字塔？对于HTTP API，如何利用测试金字塔？

你最喜欢的HTTP框架，并说明原因？

现在，我们依次来解答这些问题吧。

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。例如：

Crayon Syntax Highlighter v2.7.1.1

JavaScript

|  |  |
| --- | --- |
| 1  2  3  4  5 | fs.readFile(filePath, function(err, data) {  if (err) {  //handle the error  }  // use the data |

[Format Time: 0.0011 seconds]

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。

## 如何避免回调地域

你可以有如下几个方法：

模块化：将回调函数分割为独立的函数

使用Promises

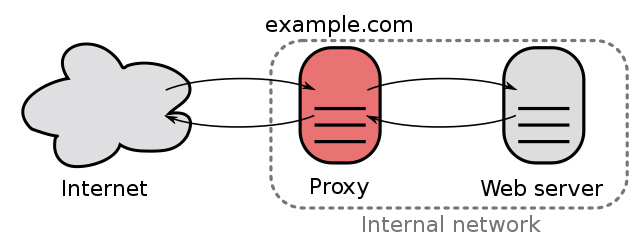
使用yield来计算生成器或Promise

解析：这个问题有很多种答案，取决你使用的场景，例如ES6, ES7，或者一些控制流库。

## 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](http://nginx.org/)）来实现，如下图所示。否则，建议你直接监听大于1024的端口。



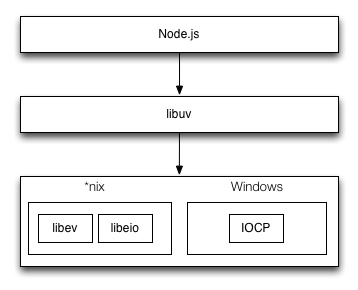
方向代理指的是以代理服务器来接收Internet上的连接请求，然后将请求转发给内部网络上的服务器， 并且将服务器返回的结果发送给客户端。

关于反向代理的更多内容，建议你阅读[这篇文章](http://www.cnblogs.com/edisonchou/p/4126742.html)。

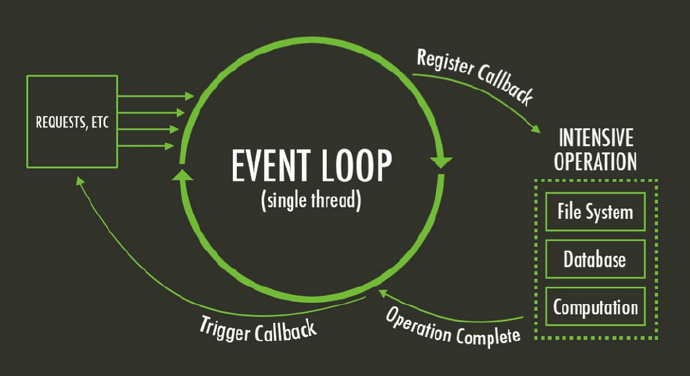
解释：这个问题用于检查被面试者是否有实际运行Node应用的经验。

## 什么是事件循环

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](https://github.com/libuv/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。 如果你需要更多详细的解释，可以参考[这个视频](https://www.youtube.com/embed/8aGhZQkoFbQ)。 你也可以参考[这篇文章](http://www.ruanyifeng.com/blog/2014/10/event-loop.html)。

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。

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](http://jslint.com/)

[JSHint](http://jshint.com/)

[ESLint](http://eslint.org/)

[JSCS](http://jscs.info/) – 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。

## 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。

## 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。

关于NPM的更多信息，你可以参考[官方文档](https://docs.npmjs.com/files/package.json)。

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。

## 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：

Crayon Syntax Highlighter v2.7.1.1

JavaScript

[Format Time: 0.0017 seconds]

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var fs = require('fs');    var readFileStub = sinon.stub(fs, 'readFile', function (path, cb) {  return cb(null, 'filecontent');  });    expect(readFileStub).to.be.called;  readFileStub. |

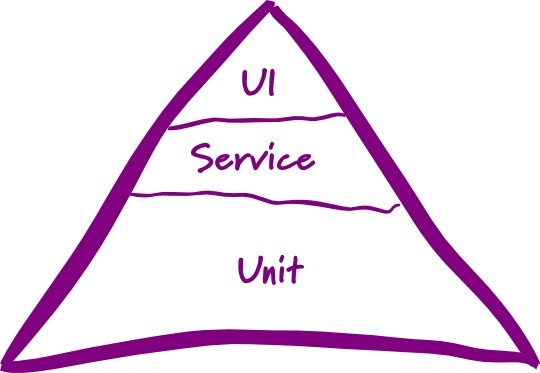
在单元测试中：Stub是完全模拟一个外部依赖，而Mock常用来判断测试通过还是失败。

## 有关Node.js的单元测试小结，你可以参考[这个链接](http://segmentfault.com/a/1190000002921481)。

解析：用于测试被面试者是否有测试的经验。如果被面试者知道什么是Stub， 那么可以继续问他是如何做单元测试的。

## 什么是测试金字塔？

[测试金字塔](http://zyzhang.github.io/blog/2013/04/28/test-pyramid/)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

解析：本文主要考察被面试者的在测试方面的经验

# 十、NodeJs

## 1、对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。

## 2、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）

## 3、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）

## 4、(如果会用node)知道route, middleware, cluster, nodemon, pm2, server-side rendering么?

Nodejs相关概念的理解程度

## 5、解释一下 Backbone 的 MVC 实现方式？

流行的MVC架构模式

## 6、什么是“前端路由”?什么时候适合使用“前端路由”? “前端路由”有哪些优点和缺点?

熟悉前后端通信相关知识

前端路由就是在不进行后端请求的情况下对页面进行跳转

## 7、对Node的优点和缺点提出了自己的看法？

优点：

1. 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。

2. 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：

1. 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。

2. 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（第三方库现在已经很丰富了，所以这个缺点可以说不存在了）。