**Nodejs面试题**

**1.为什么用nodejs，他有什么缺点？**

\* 事件驱动，通过闭包很容易实现客户端的生命活期

\* 不用担心多线程，锁、并行计算的问题

\* V8引擎速度非常快

\* 对于游戏来说，写一遍代码逻辑，前端后端通用

缺点：

\* nodejs更新快，出现版本兼容问题

\* nodejs还不成熟，还没有大制作

\* nodejs不像其他服务器，对于不同的连接，不支持线程和进程操作

-------------------------------

**2.什么是错误优先的回调函数？**

错误优先(Error-first)的回调函数(Eroor-First Callback)用于同时返回错误和数据。

第一个为参数返回错误，并验证他是否出错；其他参数返回数据

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

if(err) {

//处理错误

return console.log(err);

}

console.log(data);

});

-------------------------------

**3.如何避免回调地狱？**

\* 模块化：将回调函数转化为独立函数

\* 使用流程控制库。如[async]

\* 使用promise

\* 使用async/await

----------------------------

**4.什么是promise**

promise可以帮助我们更好地处理异步操作。下面的实例中，100ms后会打印result字符串。

catch用于错误处理。多个promise可以连接起来

new Promise((resolve.reject)=>{

setTimeout(()=>{

resolve("result");

},100);

})

.then(console.log)

.catch(console.error)

**5.用什么样的工具保证一致的代码风格？为什么要这样？**

团队协作时，保证一致的代码风格是非常的，这样团队才可以更快的维护、修改代码，而不需要每次去适应新的风格。

这些工具可以帮助我们 ：

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

\* [Standard](http://Standardjs.com/)

\* JSLint

\* JSHint

\* ESLint

\* JSCS推荐

-----------------------------------

**6.什么是stub？举例说明**

stub用于模块行为。测试时，stub可以写为函数调用返回模拟的结果。比如说，我们写文件时，实际上并不真正去写。

var fs = require("fs");

var writeFileStub = sinon.stun(fs."writeFile",function(path,data,cb) {

return cb(null);

});

expect(writeFileStub)to.be.called;

writeFileStub.resolve();

---------------------------

**什么是stub？举个应用场景**

stub适用于模拟一个组件或模块的函数或程序。在测试用例中，简单说明，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你还可以返回虚拟的结果。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.resolve();

----------------------------

**7.什么是测试金字塔？举例说明？**

测试金字塔反映了需要写的单元测试，集成测试以及端到端测试的比例：

测试HHTTP接口时应该是这样的：

\* 很多单元测试，分别测试各个模块(依赖需要的stub)

\* 较少的集成测试，测试各个模块之间的交互(依赖不能stub)

\* 少量端到端测试，去调用真正的接口(一来不能stub)

-------------------------------

**8.如何用Node监听80端口?**

\* 这题有陷阱，在Unix系统中你不应该去监听80端口，因为这需要超级用户权限，因此不推荐你的应用区监听这个端口

\* 目前，如果你一定要让你的应用监听这个端口的话，你可以通过在Node应用的前方再添加一层反代理(例如nginx)来实现，

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

------------------------

**9.什么是事件循环？**

\* 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nodejs开发者的角度是这样的。

而在底层，nodejs借助libuv来作为抽象封装层，从而屏蔽所有操作系统的差异，node可以借助libuv来实现线程。

1 nodejs----- 2 libuv---- 2.1 \*nix(libuv libeio)

|

|

2.1 Window(IOCP)

\* libuv库负责Node API的执行。他港通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。

\* 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执行

------------------------------

**10.运算错我与程序员错误的区别**

运算错误不是bug，这是和系统相关的问题，例如请求超时。或者事件故障。

而程序员错误就是所谓的bug

------------------------------

**11.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，以及指明依赖的具体版本号

----------------------