一、了解node.js

1.Node.js 是一个为实时Web（Real-time Web）应用开发而诞生的平台，它从诞生之初就充分

考虑了在实时响应、超大规模数据要求下架构的可扩展性。这使得它摒弃了传统平台依靠多线

程来实现高并发的设计思路，而采用了单线程、异步式I/O、事件驱动式的程序设计模型。这些

特性不仅带来了巨大的性能提升，还减少了多线程程序设计的复杂性，进而提高了开发效率。

2.Node.js 有着强大而灵活的包管理器（node package manager，npm）.

3.Node.js 是一个让 JavaScript 运行在服务端的开发平台，它让 JavaScript 成为脚本语言世界的一等公民，在服务端堪与 PHP、Python、Perl、Ruby 平起平坐。

4. 。Node.js 可以作为服务

器向用户提供服务，与 PHP、Python、Ruby on Rails 相比，它跳过了 Apache、Nginx 等 HTTP

服务器，直接面向前端开发。Node.js 的许多设计理念与经典架构（如 LAMP）有着很大的

不同，可提供强大的伸缩能力，以适应21世纪10年代以后规模越来越庞大的互联网环境。

5. Node.js 的 JavaScript 引擎是 V8，来自 Google Chrome 项目。V8 号称是目前世界上最快的 JavaScript 引擎，经历了数次引擎革命，它的 JIT（Just-in-time Compilation，即时编译）执行速度已经快到了接近本地代码的执行速度。Node.js 不运行在浏览器中，所以也就不存在 JavaScript 的浏览器兼容性问题，你可以放心地使用 JavaScript 语言的所有特性。

6. Node.js 最大的特点就是采用异步式 I/O 与事件驱动的架构设计.

7. CommonJS 试图定义一套普通应

用程序使用的API，从而填补 JavaScript 标准库过于简单的不足. 。CommonJS 的终极目标是制定一个像 C++ 标准库一样的规范，使得基于 CommonJS API 的应用程序可以在不同的环境下运行.

二、编译（window系统） ---没看明白

Node.js 在 Windows 下只能通过 Microsoft Visual Studio 编译，因此你需要首先安装 Visual Studio 或者免费的 Visual Studio Express。你还需要安装 Python 2（2.5以上的版本，但要小于3.0），可以在http://python.org/取得。安装完 Python 以后请确保在PATH环境变量中添加python.exe 所在的目录，如果没有则需要手动在“系统属性”中添加。

三、编程

1.运行 Node.js 程序

node script.js

node -e "console.log('hello')"；

2. REPL模式

REPL （Read-eval-print loop），即输入—求值—输出循环

在任何时候，连续按两次 Ctrl + C 即可推出Node.js 的 REPL 模式

3. 调试小技巧supervisor（英文含义：监督）

supervisor app.js 在开发 Node.js 实现的 HTTP 应用时会发现，无论你修改了代码的哪一部份，都必须终止Node.js 再重新运行才会奏效。这是因为Node.js 只有在第一次引用到某部份时才会去解析脚本文件，以后都会直接访问内存，避免重复载入

4.npm命令

npm [install/i] [package\_name]本地模式

npm [install/i] -g [package\_name]全局模式

使用全局模式安装的包不能直接通过 require 使用，但通过 npm link命令

可以打破这一限制。如，我们已经通过 npm install -g express 安装了 express，

这时在工程的目录下运行命令：

$ npm link express

./node\_modules/express -> /usr/local/lib/node\_modules/express

我们可以在 node\_modules 子目录中发现一个指向安装到全局的包的符号链接。通过这

种方法，我们就可以把全局包当本地包来使用了。

注a: npm link 命令不支持Windows。

5.调试功能先跳过

四、核心基础

1．JavaScript 的面向对象特性是基于原型的，与常见的基于类的不同。

2.utils工具

util.inherits(constructor, superConstructor)是一个实现对象间原型继承的函数。

util.inspect(object,[showHidden],[depth],[colors])是一个将任意对象转换为字符串的方法，通常用于调试和错误输出。它至少接受一个参数 object，即要转换的对象。

四个类型测试工具：util.isArray()、util.isRegExp()、util.isDate()、util.isError()

3.事件发射器

events 模块只提供了一个对象： events.EventEmitter。EventEmitter 的核心就是事件发射与事件监听器功能的封装。

EventEmitter常用的API。

EventEmitter.on(event, listener) 为指定事件注册一个监听器，接受一个字

符串 event 和一个回调函数 listener。

EventEmitter.emit(event, [arg1], [arg2], [...]) 发射 event 事件，传

递若干可选参数到事件监听器的参数表。

EventEmitter.once(event, listener) 为指定事件注册一个单次监听器，即

监听器最多只会触发一次，触发后立刻解除该监听器。

EventEmitter.removeListener(event, listener) 移除指定事件的某个监听

器，listener 必须是该事件已经注册过的监听器。

EventEmitter.removeAllListeners([event]) 移除所有事件的所有监听器，

如果指定 event，则移除指定事件的所有监听器。

EventEmitter 定义了一个特殊的事件 error，它包含了“错误”的语义，我们在遇到

异常的时候通常会发射 error 事件。当 error 被发射时，EventEmitter 规定如果没有响应的监听器，Node.js 会把它当作异常，退出程序并打印调用栈。我们一般要为会发射 error事件的对象设置监听器，避免遇到错误后整个程序崩溃。

大多数时候我们不会直接使用 EventEmitter，而是在对象中继承它，原因有两点。首先，具有某个实体功能的对象实现事件符合语义，事件的监听和发射应该是一个对象的方法其次 JavaScript 的对象机制是基于原型的，支持部分多重继承，继承 EventEmitter 不会打乱对象原有的继承关系。

4. 文件系统 fs

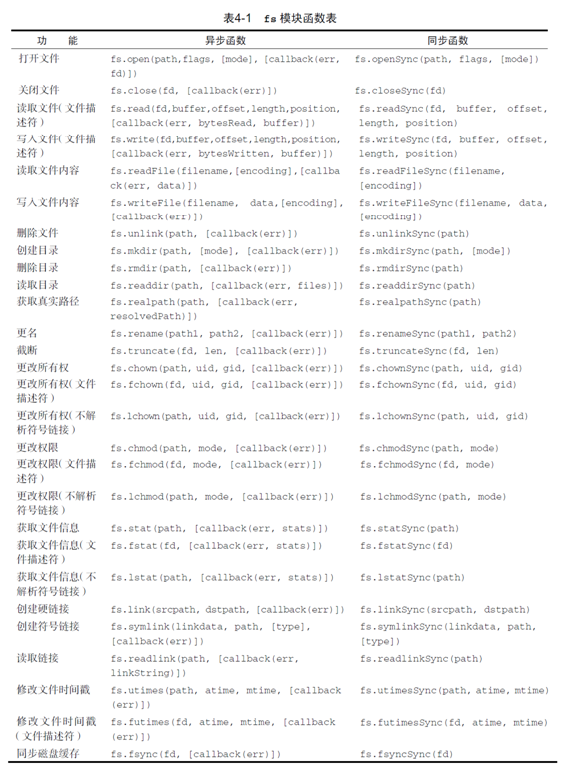
fs 模块是文件操作的封装，它提供了文件的读取、写入、更名、删除、遍历目录、链接等 POSIX 文件系统操作。与其他模块不同的是，fs 模块中所有的操作都提供了异步的和

同步的两个版本。

fs.readFile(filename,[encoding],[callback(err,data)])是最简单的读取文件的函数。（如果指定了 encoding，data 是一个解析后的字符串，否则 data 将会是以 Buffer 形式表示的二进制数据。）

fs.read的功能是从指定的文件描述符 fd 中读取数据并写入 buffer 指向的缓冲区对象.

表4-1列出了fs所有函数的定义和功能。



5.HTTP 服务器与客户端

**项目运行步骤：**

1）启动redis D:\redis64-latest>redis-server.exe redis.windows.conf

2）启动mongo D:\work\mongodb\bin>mongod.exe --dbpath=D:\work\jsgen\data

3）编译运行 F:\wudan\protect\jsgen>node app.js install

gulp

F:\wudan\protect\jsgen>node app.js recache

**第一周任务：讲一下nodejs是什么、有哪些类库、类库的特性、以及项目运行过程。**