1.使用.gitignore 忽略不需要上传的文件夹

2.package.json文件的结构

{

"name": "pos\_v2",

"version": "1.0.0",

"description": "rewrite pos\_v2 by using nodejs",

"main": "index.js",

"scripts": {

"start":"node ./main/main.js",

"test": "jasmine-node --matchall ./test/main-test.js "

},

"dependencies" : {

"jasmine" : "2.3.1"

},

"repository": {

"type": "git",

"url": "git+https://github.com/sialvsic/node-pos.git"

},

"keywords": [

"pos\_v2"

],

"author": "sialvsic",

"license": "ISC",

"bugs": {

"url": "https://github.com/sialvsic/node-pos/issues"

},

"homepage": "https://github.com/sialvsic/node-pos#readme"

}

对于当前的POS项目来说 只有上述黄线的属性有用

name：

version：

script：{

“start”: “node ./main/main.js”,

“test”: “jasmine ./test/main-test.js”

}

devDependencies：

3.使用jasmine 而不用jasmine-node jasmine-node 已过时

4.cnpm i jasmine --save-dev

i 是install的简写

此命令的作用是npm下安装jasmine 并且将jasmine安装信息写到package.json的

devDependencies的属性中去

Note：

A.安装的有cnpm 如果没有需要使用 npm install cnpm安装

B.--save 的作用是在安装的同时将安装信息写到package.json的配置中但是默认写在 dependencies中 即：软件依赖的包，其后加上-dev会安装到devDependencies中，二者并不相同详见.5

C.使用该命令安装完成，并删除一些package.json的冗余项后

{

"name": "pos\_v2",

"version": "1.0.0",

"description": "rewrite pos\_v2 by using nodejs",

"main": "index.js",

"scripts": {

"start": "node ./main/main.js",

"test": "jasmine ./test/main-test.js"

},

"author": "sialvsic",

"devDependencies": {

"jasmine": "^2.3.1"

}

}

5.构建

分为3类

production 生产

development 开发

test 测试

后两者比较相似 ，test可以算是development 的一个子集(二者在数据库测试方面有所差异)，一般情况下将测试的东西放在development中

eg：将jasmine安装在 devDependencies中

6.使用npm start命令没有问题

7.使用npm test 前 需要 npm install 会根据package.son中的软件依赖自动下载相关的依赖插件 但是我在安装完毕后会提示有出现 找不到 ../spec/support/jasmine.json的文件

解决： 本地全局安装 jasmine,然后jasmine init一下

npm install -g jasmine

8.review代码过程中的吐槽 方面：

文件结构(包括命名)： 文件的层次划分

排版：运算符之间的空格， 空行，缩进

命名：变量和函数命名

逻辑实现：

9.代码的好坏代表一个程序员的水准

10.在package.json中的diretories的作用：

[directories](https://docs.npmjs.com/files/package.json#directories)

The CommonJS [Packages](http://wiki.commonjs.org/wiki/Packages/1.0) spec details a few ways that you can indicate the structure of your package using a directories object. If you look at [npm's package.json](https://registry.npmjs.org/npm/latest), you'll see that it has directories for doc, lib, and man.

In the future, this information may be used in other creative ways.

CommonJS的规范详细介绍了几种方法，你可以指示使用的目录对象的封装结构。如果你看npm's的package.json，你会看到它有文档，lib和man目录。

在将来，这种信息可以在其他创造性的方式使用。

[directories.lib](https://docs.npmjs.com/files/package.json#directories-lib)

Tell people where the bulk of your library is. Nothing special is done with the lib folder in any way, but it's useful meta info.

[directories.bin](https://docs.npmjs.com/files/package.json#directories-bin)

If you specify a bin directory in directories.bin, all the files in that folder will be added.

Because of the way the bin directive works, specifying both a bin path and settingdirectories.bin is an error. If you want to specify individual files, use bin, and for all the files in an existing bin directory, use directories.bin.

[directories.man](https://docs.npmjs.com/files/package.json#directories-man)

A folder that is full of man pages. Sugar to generate a "man" array by walking the folder.

[directories.doc](https://docs.npmjs.com/files/package.json#directories-doc)

Put markdown files in here. Eventually, these will be displayed nicely, maybe, someday.

[directories.example](https://docs.npmjs.com/files/package.json#directories-example)

Put example scripts in here. Someday, it might be exposed in some clever way.

11.package.json script的中如果书写start和test命令

本质上是覆盖了 npm 的test和srart命令

如果在script中书写了npm中不存在的指令并eg: hehe且想要执行，那么 需要使用

npm run-script hehe

12.关于 spec和test 两者的命名差异，仅在与两者的测试驱动开发的风格不同，

spec 属于BDD下的命名

test 属于 TDD下的命名

13.package.json的作用是一致化开发环境

14.BDD与TDD

BDD 行为驱动测试

TDD 领域驱动测试

15.测试文件的命名

main-spec.js

main代表被测试文件的名称

16.入口/主文件名

not index.js main.js or app.js

17.不要在lib中写自己的代码

应该命名到main或者 src中 src中可以命名一个model