#node-lesson3

#1.复习

### 反馈

再讲下package.json的编写

3m不太清除，readfile不太清楚

可以讲快点，晚上留时间复习

## 1.0 nvm常用命令

- nvm项目地址：[nvm-github地址](<https://github.com/coreybutler/nvm-windows>)

- 查看所有已安装的本地的node版本

`nvm list`

- 切换node版本

` nvm use 版本号`

- 安装指定版本的node

` nvm install 版本号 位数`

- 卸载已安装的指定版本的node

` nvm uninstall 版本号`

###1.1模块的分类

```模块分为核心模块、文件模块，其中包是一种特殊的文件模块 ```

- 核心模块

+ fs file system

+ http

+ os

+ path

+ querystring

+ url

###1.2模块三兄弟 require、module、exports

- require的实现原理

```把代码从文件中读出来，用匿名函数的方式头尾包装，返回modules.exports对象，曝露出想要曝露出来的属性、方法、对象。

补充了exports是指向modules.exports的一个指针，exports能做的，modules.exports都能做的。

```

- require查找文件、包原理

```.js、.node、.json依次帮你补足，

第二次加载的时候优先从缓存查找读取，

node\_modules中没有的，从父文件夹中查找，如果没有，直到根目录为止

```

- module

```

里面保存了模块的信息路径、父子信息、曝露出的对象信息

```

- exports

```

module.exports 别名

module.exports=123;已经使用过一次了，exports就不生效了

module.exports.a=12;这样使用,exports是还可以使用的

```

###1.4包

```

package.json是包的说明书，

dependencies当前包所依赖的包：>、<、>=、<=、~、\*、" "、X、^、

devDependencies 开发环境中包所依赖的包：

```

###1.5npm

```

npm init 创建package.json文件

npm install 将package.json中的文件依赖的包从网上下载到本地

npm install -save 包名 将包下载下来并且加载到dependencies中去

npm i -S 包名 替代上面的命令

npm install -save-dev 包名 将包下载下来并且加载到devDependencies中去

npm i -D 包名 替代上面的命令

npm install -g 包名 全局安装 安装的是全局工具

npm docs 包名 查看包的文档

```

###1.6nrm npm的数据源管理工具

```

nrm ls 查看npm的数据源

nrm use 切换npm下载包的地址

nrm test 测试下哪个数据源快

```

### 1.7 3m

npm node的包管理工具

nvm node的版本管理工具

nrm npm的数据源管理工具

###1.8箭头函数

```

语法糖替代function(){} v=>v

```

#2.i/o

###2.1什么是i/o？

```

io input、output 输入输出，电脑的输入输出，例如音频录音表示声音输入、听音乐是声音的输出

网络上的传输全部是在传字符串，i/o在服务器上可以理解为读写操作。

```

###2.2什么是并发？

```

一个时间段中有几个程序都处于已启动运行到运行完毕之间。

```

#3异步i/o与事件驱动

###3.1什么是进程？

```

进程是为运行当中的应用程序提供运行环境的

一个运行当中的应用程序就会有一个进程与之相对应

```

###3.2什么是线程？

```

线程是用来运行应用程序中代码的，

一个线程在一个时间只能做一件事件。

多线程，调度起来很麻烦。

node是单线程执行，用异步替代了多线程

```

###3.3同步、异步有什么不同？

```

异步不会阻塞后面的代码，同步会阻塞后面的代码

一条线程先执行同步的代码后执行异步的代码。

```

node中异步就是帮你完成多线程编程。node中是没有让开发者手动开启新线程的API的，只有node指定的一些异步操作，会默认的自带开启新线程的功能。

node指定的异步操作：

1异步非io setTimeout() setInterval()

2异步io 文件操作

多线程编程比较复杂

```

###1.3同步和异步？

```

同步会阻塞代码

异步不会阻塞代码

```

###1.4事件驱动模型

```

主线程：

1.执行node的代码，把主线程中的代码放入队列

2.事件循环程序（主线程）把队列里面的同步代码都先执行了，

3.同步代码执行完成，执行异步代码

4.此时队列是并行队列的性质，队列中异步的任务们同时执行。首先去可调度线程池中，找有没有现成的，如果没有就创建新的线程。

5主线程会一直轮询，队列中没有代码了，主线程就会退出。

子线程：被放在线程池里面的线程，用来执行异步io操作

1.在线程池里休息

2.异步io的操作来了，执行异步io操作。

3.子线程会把异步io操作的callback函数，扔回给队列

4.子线程会回到线程池了去休息。

callback

在异步io代码执行完成的时候被扔回主线程。下面排队执行。因为子线程中执行任务的快慢不一，所有callback回来的顺序也是不一样的。被主线程执行的顺序也不定。

###3.4异步非i/o操作和异步i/o操作

```

异步非io setTimeout setInterval

异步IO操作 操作文件 网络操作 fs

```

###3.5node的事件驱动模型？

![Alt text](./pic/event-loop.png)

```

```

###3.5异步和多线程的比较？

```

node的异步是帮助我们去做了多线程的操作，简化了代码

```

#4.文件操作

###4.1文件的完整读写 文件读写属于异步io操作，所以一般需要回调: 箭头函数

```

fs模块---》操作文件---》io----》node的特长

fs模块是node非常重要的模块，能体现出node的优势

```

- fs.readFile() 读文件 一般都是utf8格式

- fs.writeFile() 写文件

- fs.appendFile() 在文件的内部去追加写一些内容

- fs.mkdir() 创建文件夹

- fs.readdir() 读文件夹，只读子文件/子文件夹，不向下

- fs.access() 判断路径 不光可以判断文件，还可以判断文件夹

- fs.stat() 同上，并增加了成功回调的一个参数stats对象，这个对象有很多方法，可以判断是文件，还是文件夹，还有属性，文件大小，创建时间等

isFile：用于判断被查看的对象是否为一个文件，如果是返回true，否则，返回false；

isDirectory：用于判断被查看的对象是否为一个目录，如果是的话则返回true，否则，返回false；

isSymbolicLink：用于判断被查看的文件是否为一个符号链接文件，如果是的话则返回true，否则，返回false。该方法仅在lstat方法的回调函数中有效；

路径，是相对于调用readfile方法的地方，不是我们封装readfile方法的路径目录

不要同时操作一个文件

#5.异常处理

什么是异常？

计算机不能处理的状况称之为异常

nodejs代码，同步代码，如果执行结果错了，会在命令行中打印错误信息，如果是异步的执行结果的回调error是不会打印在控制台的。

try catch 只能捕获同步异常，因为try catch是同步代码，异步里的执行结果的error出来的时候，try catch早过去了，所以异步里的error只能手动throw跑出来

try {

fs.accessSync('./02sfdasdfio.js');

} catch (err) {

console.log("1"+err);

}

fs.access('./0hhh2.js', (err,data)=>{})

异步异常对象和被try catch捕获的error对象是一样的

语法上的错误，命令行控制台中都会打印出来

throw抛出异常的时候，代码执行就会停在这，不会向下执行

nodejs的方法api有的解释很少很少，不是开发人员懒，是因为nodejs上的api，与linux的命令作用是一致的，想对应的。比如fs.readfile解释很少，直接搜linux: readfile命令什么意思，再比如makedir。注意别搜fs.readfile方法，搜命令行

一些逼格高的公司，让你直接用linux开发，把windows卸了

#6.path

path原生内置对象的api