#node-lesson4

# 反馈

同步代码和异步代码分不清

异步非io模型图不是很理解

path使用有点不太清楚

知识点多，比较乱，需要整理

#1.复习

###1.1什么是io?

```

io 是输入输出 文件操作的读写 网络操作中的请求和应答

```

###1.2进程和线程？

```

进程是为运行中的应用程序提供运行环境的

线程就是执行应用程序当中的代码的

同一时间只干一件事情

###1.5文件操作

```

- fs.readFile() 读文件

- fs.access() 判断路径

- 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.stat() 判断文件文件夹，文件/文件夹的属性

- fs.readdir()

```

迭代 f(1),f(2),f(3),f(4)......

递归 f(f()),f(f(f())),f(f(f(f()))).....

###1.6path path.join("/a","/b",,,,,)

#1.7 $http

$http(

{

method: 'GET',//发送get请求

url: 'http://127.0.0.1:3000/in\_theaters/'

}

).then(function(data){},function(error){})

.then的成功方法里面返回的data是整个报文，如果取里面json数据需要data.data

.success的方式去取的data是报文体，data就是数据

.then的执行过程，不成功，就接着再来一遍，详情见照片

#2.node的调试

###2.1 node自带的debugg

###2.2 node-inspector第三方的调试包

###2.3 vsc 调试

###2.4 ws 调试

###2.5异常处理

error .message .stack .code

var err =new Error('这是一个自定义错误');

# 其它

- vsc教程[http://i5ting.github.io/vsc/](vsc教程)

#3.es6

###3.1什么是es6？(又叫es2015，es7是2016)

```

ECMAScript es标准的作用 定义js的发展方向的一套标准

```

###3.2为什么用es6？

```

新的语法，写起来更严谨更安全更简便。

```

###3.3为什么前端不用，node要用？

```

js--->浏览器-->引擎执行js代码

ie ff chrome

node---->v8执行js代码

node 不考虑兼容性，es6能给我们带来很多好处。

Node.js 6覆盖了93%的ECMAScript 6

```

前端开发中有的还在用es3 4呢

- [阮一峰es6入门](http://es6.ruanyifeng.com/#README)

###3.4 use strict严格模式

```是es5引入的，使用了以后，运行的时候，会按照最新语法，来运行，以前的一些不严谨的语法，严格模式的时候会报错，比如: a = 5; 这种表达式严格模式就会报错

总结: JavaScript语法在慢慢的变严谨，变完善，越来越向一门真正的开发语言语法进化(JavaScript刚开始设计的时候只是用来做一些小功能，并不是作为一门真正的编程开发语言设计创建的)。向java，oc，swift这种完整语言进化

消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;

　　- 消除代码运行的一些不安全之处，保证代码运行的安全；

　　- 提高编译器效率，增加运行速度；

　　- 为未来新版本的Javascript做好铺垫。

'use strict'

保留字：

一旦出现Unexpected strict mode reserved word这样的错误说明你用保留字做了参数名了。

es6-->implements, interface, let, package, private, protected, public, static, yield

es5-->class, enum, export, extends, import, super

```

[阮一峰 严格模式](http://www.ruanyifeng.com/blog/2013/01/javascript\_strict\_mode.html)

打开这个博客，挺全的，老师就照这个讲的

###3.5 let

在使用let命令声明变量时，一个变量名只能声明一次，不存在命名冲突。

let定义作用域在代码块里,解决命名冲突问题，能体现代码的封装性

变量提示，用let不存在变量提示，会直接报错，告诉你变量没有定义

let命令每次定义的作用域都在本次循环的方法体内

解释有点模糊，自己上网查查？

###3.6 const

//定义常量，不会发生变化

//如定义了常量，再去修改就会报错error Assignment to constant variable 不可以给常量赋值

//const定义的常量，作用域与let相同

//应用场景const用来定义静态变量,加载模块的时候，定义个常量，把模块赋值给常量

###3.7 块级作用域

凡是被{}包裹的代码属于一个代码块

###3.8promise

https://cnodejs.org/topic/560dbc826a1ed28204a1e7de

异步流程控制的解决方案

回调地狱

.then()

异步流程--->回调函数的嵌套

异步流程控制---->用同步的方式去写异步的代码

###3.8 字符串的一些扩展`${}`

str.includes(str) 是否包含str这个字符串

str.startWith(str) 是否以str开头

str.endWith(str) 是否以str结尾

str.repeat(次数)： 循环打印字符串

* 模板字符串``
  + 增强版的字符串：用反引号(`)作为标识
  + 模板字符串中所有的空格和缩进都会被保留
  + 在模板字符串中嵌入变量：${变量名},可以有多个
  + 模板字符串可以是原始的：String.raw`hello world\n`

## 4. 文件操作

### 4.1 Buffer

是一个数组对象，每个元素存放两位十六进制数字。文件流！

十六进制：0、1、2、3、4、5、6、7、8、9、a、b、c、d、e、f 满16进位 所以 0X10=16

计算机最早诞生的时候，没有中文，在美国

26个英文字母，@ ! , . - + =

计算机只能识别 0 或者 1

把你的字符和计算机真正存储的二进制数据做了一个字典：

可以通过 开源中国 官网网站提供的一个工具页面：http://tool.oschina.net/

随着计算机普及到了世界各地，ASCII 码 已经不能满足世界各地人们的需求了

计算机进入中国之后，后来在原来的 ASCII 码基础之上扩展了一个新的字符集编码：gb2012，

字典 大字典utf-8

用来支持中文

字节的转化

1汉字=2字节 00100010 00000000

1字节（Byte)＝8字位＝8个二进制数

1字位(bit)＝1个二进制数

1B=8b

1KB=1024B

1MB=1024KB

1GB=1024MB

#### 4.1.1 创建Buffer

Buffer 是一个像 Array 的对象，它的元素为16进制的两位数（0-255的值），主要用于操作字节，

Buffer 是一个全局对象，使用的时候不需要 require

- new Buffer(size)

- new Buffer(str,[encoding])

#### 4.1.2 Buffer的一些属性和方法

- buf[index] 通过下标访问 buffer 的某个字节的数据

- buf.indexOf(value,[byteOffset],[encoding]) 查找某个字符在 buffer 内存中的字节下标

- buf.includes(value,[byteOffset],[encoding])

- buf.length

- buf.slice([start,[end]])

- buf.toString([encoding],[start], [end])

- buf.write(string,[offset],[length],[encoding])

#### 4.1.3 造成乱码的原因

我们通常所说的编码一般就是指 字符集编码，一般用于字符串

当你把一个 `gbk` 编码文件发送给了你的一个国外友人的时候，

乱码的原因就是：他的机器上没有该编码

计算机为了让多语言操作系统下可以识别和共享一种编码格式：所以诞生了 超级大字典：`utf-8`

造成乱码的原因其实就是　读取的编码和文件的编码不一致

要想解决乱码：让文件编码和读取编码统一即可

- 什么是字符集编码

- 为什么要有编码

+ 计算机只能识别二进制

+ 为了让计算机可以识别字符，人类做了一个字典 二进制 -> 字符 的映射关系

- 为什么会产生乱码

+ 文件编码和读取该文件的编码不一致导致的

- 如何解决乱码

+ 让文件和读取的字符编码集一致即可

- 如何解决 Node 原生不支持的一些编码

+ 通过 第三方包：[iconv-lite ](https://www.npmjs.com/package/iconv-lite)

+ 该第三方包可以解决 gbk 等编码不支持的问题

###4.2文件流

- 流对象 stream

- fs.readFile()和fs.writeFile()问题？

+ 对大文件的处理，例如下载

- 通过文件流的形式传输大文件

const rs = fs.createReadStream(path1);

const ws = fs.createWriteStream(path1);

rs.pipe(ws);

-如何去控制流。

我们可以通过监听stream对象里面的事件，去控制读写流。

rs有data end事件 直接用on绑定就行，前者是传输过程中，每读一小块文件就调用，回调中有一个参数，每次读到的一小块文件的大小

# 其它

- vsc教程[http://i5ting.github.io/vsc/](vsc教程)

# 作业

- 装node-inspector

- 读http://es6.ruanyifeng.com/#README

- vsc教程[http://i5ting.github.io/vsc/](vsc教程)

nodejs调试: sublime vscode webstrom

nodejs中的api，看一下官方文档api

es6新增的语法，看一下阮一峰的那个博客