# node-lesson2

# 1.复习

## 反馈

系统全局变量有什么用还不是很清楚。

讲的有些快有些跟不上

## 1.1 概念

- Node.js与JavaScript的关系?

- Node.js基于哪个JavaScript引擎？

- Node.js的特点？

- Node.js只能运行在Windows上吗？

## 1.2 cmd

- cd（change directory）切换目录

- md（make directory）新建目录

- rd（remove directory）删除非空目录

- dir（directory）查看目录中的条目

` ls linux`

- ren（rename）重命名文件

- del（delete）删除文件

- cls（clear screen）清屏

`clear linux`

## 1.3 path 环境变量

- path环境变量的作用是什么？

- 在Windows中配置path环境变量的方法是什么？

## 1.4 REPL（Read Eval Print Loop）交互式运行环境

- 作用：一般来说，可以用来做一些API的测试

- 进入REPL运行环境

- 退出REPL运行环境

## 1.5 Node.js基础知识

- 如何让Node.js执行一个js文件

`node 文件名`

-绝对路径和相对路径

`.\或者..\ 相对路径`

`c:/xx/xx.js绝对路径`

`\ linux 根目录

windows 盘符的根目录`

## 1.6 globals

- global

- \\_\_dirname和\_\_filename

` 关于这俩家伙，一般在读取文件的时候，最好使用绝对路径的方式，通过这俩家伙拼接 `

- console

+ assert(foo==1,"失败");断言

+ time('timer')

+ timeEnd('timer')

+ 成对的出现

## 1.7 模块系统

- 什么是模块

`一个文件就是一个模块 `

- 模块作用域

`在一个模块内变量、函数、对象都属于这个模块，对外是封闭的。 `

- require

`require("路径")`

//路径现阶段使用相对路径

- exports

`是module.exports的一个别名、引用，exports能做的module.exports都可以完成。`

- module.exports

`最终曝露出去的对象，想曝露出去对象、属性、方法就挂载到module.exports`

#2.模块系统

##2.1模块的种类

在Node.js中，模块分为两类：

```

第一类，核心模块（原生模块），node自带，用名称直接可以加载。

- 核心模块

+ fs file system

+ http

+ os

+ path

+ querystring

+ url

...

第二类，文件模块，用路径加载，有一种特殊的文件模块----->包，可以用名字。

```

module 是一个全局对象，它的作用是存储模块信息的,每一个js是一个模块，每一个模块里面都有一个module，module里面还存储了父子结构

直接给module.exports赋值，exports是无效，用属性的形式去赋值的时候二者都有效

## 2.2 优先从缓存加载

```

common.js规范----->加载后，再次加载时，去缓存中取module.exports

```

## 2.3 require参数解析

```

核心模块是Node.js原生提供的加载核心模块的时候，

不需要传入路径，因为Node.js已经将核心模块的文件代码

编译到了二进制的可执行文件中了

在加载的过程中，原生的核心模块的优先级是是最高的

```

![Alt text](./pic/image1.jpg)

- ./或../开始的相对路径文件模块

``` 在加载一个自己编写的模块的时候，最好使用\_\_dirname 和 你要加载的模块的文件名拼接 ```

- 以/开始的绝对路径文件模块

```

\* 在Linux或者MAc的操作系统中，/表示系统的根路径

\* 在Windows中，/表示当前文件模块所属的根磁盘路径

不建议使用，了解

```

- 标识符中可以不包含扩展名

+ Node会按照`.js、.node、.json`的次序补足扩展名，依次尝试

```

不包含扩展名的时候，Node.js会按照

路径.js 以后自己在加载js文件模块的时候，就省略掉.js后缀就可以了

路径.node 后缀为node的文件是c/c++写的一些扩展模块

路径.json 如果是加载json文件模块，最好加上后缀.json，能稍微的提高一点加载的速度

.json文件最终Node.js也是通过fs读文件的形式读取出来的，然后通过JSON.parse()转换成一个对象

Node.js会通过同步阻塞的方式看这个路径是否存在

依次尝试，直到找到为止，

如果找不到，报错

```

- require 加载包的时候

```

直接写包名，先在包内的node\_modules,

目录下查找，去父级目录下的node\_modules目录下查找，

依次向上查找，直到根目录。

```

[参考文献](http://www.infoq.com/cn/articles/nodejs-module-mechanism/)

# 3.包

还有一种特殊的文件模块，其实就是包

- 问题：你开的一个功能模块，还是以文件的形式散列的，你给别人使用的时候很麻烦。

- 包的作用：在模块的基础上进一步组织JavaScript代码

## 3.1 包的组成

- 包结构

- 包描述文件package.json

+ 描述你的当前的包的一些相关的信息

+dependencies

- 包名：“版本号”

- > +版本号 下载大于某个版本号，npm会下最新版

- < +版本号 下载小于某个版本号，npm会下小于这个版本号最新版

- <= 小于等于 一定会下你写的这个版本，除非没有你写的这个版本

- >= 大于等于 下载最新版

- \*、" "、X 任意 npm会给你下最新版

- ~ +版本号 会去下约等于这个版本的最新版，在大版本不变的情况下下一个比较新的版本

- ^ +版本号 不跃迁版本下载，^2.1.0 npm会下载大版本不变，去下载2.x.x版本里的最近版

[参考文献 package.json全字段解析](http://blog.csdn.net/woxueliuyun/article/details/39294375)

##

npm install -g fast-init

## 3.2 package.json

包的描述文件

package.json文件内部就是一个JSON对象，该对象的每一个成员就是当前项目的一项设置，

比如name就是项目名称，version就是项目的版本号

在模块的加载机制中，有一个main属性是非常重要的，它很大意义上决定了包要导出的模块位置

## 3.3

# npm

- 基于Node.js开发的包的托管网站

- Node.js包管理工具

npm :基于包的规范实现的一个包管理工具

- npm install 包名

当执行npm install的时候，它会自动跑到npm的网站，然后找到该包的github地址，

找到之后，下载这个压缩包，然后在执行npm install的当前目录下找一个叫做node\_modules目录

如果找到，直接解压这个压缩包，到node\_modules目录下

如果找不到，则新建一个node\_modules目录，解压到该目录

- npm install -g 包名

表示全局安装

npm在安装的时候，在全局设置了一个安装目录，只要是全局安装就会把包安装到这个目录里面去

npm install -g 包名 只要是全局安装就是安装命令行工具

- npm install 缩写 npm i

当执行npm install的时候，会自动在当前目录中查找package.json文件

如果找到，找里面的 dependencies 字段，安装该字段中所有依赖的项

- npm docs 包名

通风这个命令打开 包相对应的文档

- npm install --save 生产环境

npm i -S jquery@3.\*

咱们以后在做项目的时候，先初始化一个package.json文件，

在安装第三方包依赖的时候，必须使用npm install --save express,添加依赖项到package.json文件中，

实际就是添加到dependencies字段中

- npm install -save-dev 开发环境

npm i -D jquery@3.\*

咱们以后在做项目的时候，先初始化一个package.json文件，

在安装第三方包依赖的时候，必须使用npm install --save express,添加依赖项到package.json文件中，

实际就是添加到devDependencies字段中

- npm config set prefix 修改全局安装目录

不建议使用，建议去文件夹里面改 C:\Users\用户名 里面有一个 .npmrc文件

去里面改prefix = C:\dev\npm （写你的路径）

去全局的环境变量path里把C:\dev\npm 路径加进去

[参考文献 npm的命令](https://docs.npmjs.com/)

# npm 这种东西的最终的目的就是：让你的开发模式工程化，都依靠工具来管理

# nvm npm nrm

## 3m

```

nvm node版本的管理工具

npm node的包管理工具

nrm npm的数据源管理工具

npm install -g nrm

```

#4.文件操作

###4.1箭头函数

```

箭头函数是ES6才出的语法，开发者工具的设置里，注意调一下兼容语法的上限版本，否则，代码会保错

箭头函数就是一种语法糖

语法糖是一种语法，用这种语法能尝到甜头，能是编程高效

当函数体有一个参数有返回值的时候

var foo=function(v){ return v;}

var foo=v=>v;

当函数体没有参数有返回值的时候

var foo=function(){ return v;}

var foo=()=>v;

当函数体有多个参数有返回值的时候

var foo=function(v1,v2){ return v1+v2;}

var foo=(v1,v2)=>v1+v2;

当函数体有多个参数没有返回值的时候

var foo=function(v1,v2){

console.log(v1); console.log(v2);

}

var foo=(v1,v2)=>{

console.log(v1); console.log(v2);

};

```

###4.2文件读写

- fs.readFile() 读文件

fs.readFile(file, [options], callback)

//file 读文件的路径

//options 用[]包裹的可选参数，object {"encoding":"uft8","flag":"r"} string 'utf8'

//callback function callback函数里面有两个参数

//err表示错误对象，如果readFile发生错误了，在callback函数里面会回返回err，err对象里面存着错误信息

//如何读文件成功，err为null

//data就是读文件，读出来的内容。

- fs.access(path,[mode], callback) 用来判断文件是否存在

//path 判断路径是否正确（也可以判断文件）'./pic'

//mode一般不用，默认就可以了，用的话是用来判断文件是否可读可写可执行

//callback 只有一个参数err，错，文件有问题

# 其它

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

#作业

2.npm 和package.json结合着连

npm init 初始化package.json

npm install 先配置package.json里的devDependencies dependencies字段

npm i -S +包名 npm install –save 不光能下载包，还能把依赖添加到package.json

npm i -D +包名 npm install –save-dev 不光能下载包，还能把依赖添加到package.json

npm docs 包名 查看包的相关文件 jquery、mongoose、http-poster

npm list 当前目录有什么包，检测node\_modules里面有什么

npm install –g 包名 全局安装 cnpm、nrm

3.cnpm

4.nrm

安装目录npm install -g nrm

nrm ls 查看数据源

nrm test 测试你的网连那个npm的数据源快

nrm use 切换数据源

5.阅读参考文献 朴灵 require

阅读参考文献 package.json全字段解析

6.github自己玩玩

$interval在ng中的作用检查机制

setInterval是不起作用的