# 2018.03.18

## 什么是Node.js

NodeJs是运行在服务器端的JavaScript，有服务器端的功能：文件系统，模块，包（NPM）可以与数据库交互，开发基于服务器管理的后台管理系统（如：CMS，CRM，OA，ERP）。

## Node.Js和JavaScript的区别

Node.Js = JavaScript执行环境+JavaScript扩展功能

NodeJs 是运行在Chrome V8的JavaScript引擎。Node.js使用一个事件驱动，非阻塞式I/O模型。包管理器NPM.

NodeJS 运行在服务器端，需要服务器的支持

JavaScript 运行在客服端，只需要浏览器的支持

相同点：ECMAScript的基础语法一致。

## node.js和V8的关系

查资料

<https://kb.cnblogs.com/page/573533/>

编程语言分为编译型语言和解释型语言两类，编译型语言在执行之前要先进行完全编译，而解释型语言一边编译一边执行，很明显解释型语言的执行速度是慢于编译型语言的，而JavaScript就是一种解释型脚本语言，支持动态类型、弱类型、基于原型的语言，内置支持类型。鉴于JavaScript都是在前端执行，而且需要及时响应用户，这就要求JavaScript可以快速的解析及执行。

随着Web相关技术的发展，JavaScript所要承担的工作也越来越多，早就超越了“表单验证”的范畴，这就更需要快速的解析和执行JavaScript脚本。V8引擎就是为解决这一问题而生，在node中也是采用该引擎来解析JavaScript。

渲染引擎：能够将HTML/CSS/JavaScript文本及相应的资源文件转换成图像结果。渲染引擎的主要作用是将资源文件转化为用户可见的结果。在浏览器的发展过程中，不同的厂商开发了不同的渲染引擎，如Tridend(IE)、Gecko(FF)、WebKit(Safari,Chrome,Andriod浏览器)等。WebKit是由苹果2005年发起的一个开源项目，引起了众多公司的重视，几年间被很多公司所采用，在移动端更占据了垄断地位。更有甚者，开发出了基于WebKit的支持HTML5的web操作系统(如：Chrome OS、Web OS)。

## node.js的应用

web后台开发的编程语言：JSP PHP ASP.net nodejs

### node.js优势

node.js基于JavaScript，学习门槛低

前后端分离（前后端的服务器分工协作，互不干扰）的开发思想，主流开发方式。（不分离：同域；分离：跨域）

前端开发工具很多基于Node.js 有庞大的生态圈，生态链（GitHub.com）

### node.js下载安装

<http://nodejs.cn/>

### node.js运行方式

方式1： 进入node.js程序的目录：在运行文件下，调用cmd，输入node filename.js (.js可以省略)

方式2： 在webstorm的terminal的控制台输入（同方式1）

方式3： 在webstormde上直接运行（中文也可以打开到）

### node.js控制台

console.log()

console.error() 有颜色标记

console.time(‘name’)

console.timeEnd(‘name’)

console.assert()

断言，用于代码中捕获假设，可以看做是try catch finally的一种高级形式

断言就是一种错误抛出机制，如果表达式为假，则抛出错误，为真不执行

console.dir() 以对象/目录的形式展开

## 模块化

根据项目的需求，把整体功能分解为多个子功能，每个子功能就是一个模块。把每一个模块的开发分配给不同的开发人员。

目前流行的的模块规范：AMD CMD Commonjs(Node.js)

JS模块的共性，模块的功能都是在一个函数中：

1. 模块中var定义的都是局部变量
2. 模块定义在函数中，是局部
3. 模块对象，
4. 模块中需要暴露，

asychronous module definition异步模块定义，这种规范是异步加载模块 ---代表 requireJs

common module definition公共模块定义，国内大牛玉伯之作，---代表sea.js

CommonJS 只是标准，主要是对JavaScript标准API进行增强，---代表node.js

### 模块的使用

模块：一个具有特定功能的文件就是一个模块，模块之间可能存在一个依赖关系

在子模块中定义的变量和函数，通过exports.newName = moduleA;来输出模块

使用子模块，需要var callname = require(‘./ ../ ../../ /’); callname.newName

require的语法：

var newName = require(‘path of module’)

注意：扩展名是.js .json .node

exports的语法：

exports.函数名 = 函数引用；

exports.变量名 = 变量名；

### 模块函数

由于node.js是基于模块化开发，所以每一个模块最终都会封装在一个匿名函数，每个文件就是一个模块。

*function (exports, require, module, \_\_filename, \_\_dirname) {*

*。。。。 自定义内容  
 }*

exports：暴露对象

require：引入方法

module：模块对象（包含了模块自身一些属性和方法，其中有exports，path）

\_\_filename: 文件的完整路径，包含文件

\_\_dirname: 文件的完整路径

### 模块对象的使用（推荐此方式，而非exports）

module.exports.函数名=函数的引用；

module.exports.变量名 = 变量名；

module.exports=函数/变量；

### exports和module的区别

两者几乎一样

exports是对module.exports的引用，所以只能在exports上面添加属性和方法，不能替换它

module.exports才是真正的暴露对象，指向哪里就暴露哪里

module.exports可以暴露任意类型数据，可以重写

### 模块的分类

三种模块：

系统模块：官方人员开发的模块，只要有nodejs环境，就可以直接使用，比如：http/fs/url

var fs = require(‘fs’); //引入文件系统

第三方模块：使用NPM安装模块名或者包名，安装好之后，会自动放入node\_modules文件夹下

安装：npm install --save multer

使用：var multer = require(‘multer’);

自定义模块：先暴露再引入

## 作用域

规定了一个变量和函数可使用范围。全局作用域，局部作用域

全局：

gobal.c

a

局部：

var b

为了避免变量在主模块中出现冲突

# 2018.03.19

## NPM

Node Package Manager 包管理工具（第三方模块），可以下载，安装，上传，以及管理已经安装好的包。

[www.npmjs.com](http://www.npmjs.com)

## package.json

它是包描述文件（主模块文件，依赖，作者，版本，贡献者。。）可以使用npm.init(需要逐个输入内容)/npm.init -y（自动生产）得到

name：包名

version：包版本

description：包描述

main：入口文件（主模块）

homeage: 官网网址

contributors：贡献者

dependencies：生产阶段依赖包列表，如果依赖包没有安装，npm 会自动将依赖包安装在 node\_module 目录下

devDependencies: 开发阶段依赖包列表

script：命令的名称--可以通过npm start commandName启动此命令

git repository ：git仓库地址

keywords:关键字

## npm常用命令

npm init -y 初始化生成一个package.json

npm install packagename (可以简写为i)

npm i jquery@2.24（@后面跟版本号）

参数：

-g 全局安装 global

--save -dev 或者 -D 添加到开发阶段依赖列表

--save 或者 -S 添加到生产阶段依赖列表

全局安装nodemon模块（支持热刷新看结果）

npm i nodemon -g -D

卸载：

npm uninstall packagename

## CNPM：

NPM服务器在美国，在国内下载很慢。

CNPM是taobao提供一个完整npmjs.org镜像，cnpm和npm的内容每10分钟会同步一次

安装CNPM

npm install -g cnpm --registry=https://registry.npm.taobao.org

使用CNPM

cnmp install packagename

## NodeJs回调函数

简称回调（callback）就是把一个函数作为参数传递给另一个函数

回掉函数的用途

当一个事件或者任务完成以后再去执行回调函数的内容，最终输出结果.

## events事件编程

大多数Node.js核心API都采用惯用的异步事件驱动框架，其中某类型的对象（触发器）会周期性地触发命名事件来调用函数对象（监听器）。

1. 引入events模块 const events = require(“events”);
2. 实例化eventEmitter类 var e = new events.EventEmitter();
3. 注册事件 e.on(‘name’,callback())
4. 触发事件emit()方法 e.emit()

eventEmitter()对象常用方法：

.on(‘name’,callback())

.addListener(‘name’,callback())等价于on

.once(‘name’,callback())只能触发一次

.setMaxListeners()最大数量的监听数量（0代表无限制，可能导致内存泄漏;需要在注册之前设置，否则无效）

.emit(‘name’,agr1,agr2....)触发事件

.removeListener(‘name’,callback())移除事件监听

.removeAllListener(‘name’)

## http

hyper-text transfer protocol超文本传输协议：是互联网上应用最为广泛的网络协议

hyper-text transfer protocol safe 超文本安全传输协议

localhost本地主机

127.0.0.1本地IP

182.139.173.24 局域网IP地址

http模块

访问任何一个网站都是一个请求request和响应response的对象

## httpServer

const http = require(‘http’);

var visitors = 0; //记录访问人数

const server = http.createServer((req,res)=> {

//解决编码问题

res.writeHead(200,{

“Content-Type”:”text/html;charset=utf-8”

})

console.log(‘服务器端 控制台看到的内容’)

visitors++; //每次响应增加一个访问

res.write(‘客服端得到的响应内容’+visitors);

//获得IP

var ip = res.socket.remoteAddress.replace(‘::fff:’,’’);

res.write(‘您的IP地址是’+ip);

res.end(); //响应需要有结束，否则一直响应着

});

server.listen(8080,()=> {

console.log(‘服务器已经启用‘);

}) //可以使用链式调用

总结：

引入HTTP模块

创建服务器 HTTP.createServer((req,res)=>{})

监听服务器端口server.listen(port)

## http.get()

大多数请求是get请求，不带请求主体，所以nodejs提供了便利的方法

get(options[,callback])

options:obj string URL

[,callback]监听事件类型:

‘data’:有数据流入

‘end’:数据响应结束

‘error’:发生错误

const http = require(‘http’);

http.get(‘https://www.baidu.com’,(res)=>{

res.on(‘data’,(html)=>{

console.log(html.toString()); //爬虫技术需要

});

res.on(‘end’,()=>{

console.log(‘加载完毕’);

})

res.on(‘error’,(err)=>{

throw err;

})

})

## promise()

### 同步与异步

同步：

一个任务等待前一个任务结束，然后再执行后一个任务，程序的执行由前向后。

异步：

每一个任务有一个或者多个回调函数，前一个任务结束后，回调函数的结果不会立即执行，程序的执行顺序与任务的排列顺序不一致，是异步的。

### 异步编程实现方法：

回调函数

事件注册

promise：就是一个对象，用来控制异步操作的结果

三个状态：

pending:进行中

resolved:已完成

rejected:已失败

状态改变后不能再变，pending--》resolved pending--》rejected

var promise = new Promise((resolved,rejected)=> {

if (true) {

resolve(value);

} else {

rejected(error);

}

});

//单任务

promise,then((value)=>{

//success

} (error)=>{

//failed

}),

//多任务

promise.all([task1,task2]).then((result)=>{

console.log(result[0]);

console.log(result[1]);

},(err)=>{

throw err;

})

# 2018.03.21

## Buffer缓冲区

### 储存单位

byte叫做字节，由8个bit组成，用于表示计算机中的一个字符。

8bit = 1byte

1024byte = 1kb

1024kb = 1mb

1024mb = 1g

1024g = 1t....

### buffer存在意义

JavaScript语言自身只有字符串数据类型，没有二进制数据类型。

node.js操作文件流时需要使用二进制，所有定义了一个Buffer类，来创建一个专门存放二进制数据的缓冲区。

### 创建buffer

1. new Buffer(字节的大小)

var buffer1 = new Buffer(20);

buffer1.toString

1. new Buffer(ASCII编码数字以数组的形式)

var buffer2 = new Buffer([65,66,67,68,69]);

buffer2.toString //ABCDE

1. new Buffer(‘字符串’，‘编码方式’)

var buffer3= new Buffer(‘随便’，‘utf-8’);

buffer3.toString //随便

### 写入buffer

buffer.write（‘内容’，‘编码’）

var buffer4= new Buffer(200);

buffer4.write(“asp.net C#”,”utf-8”); //因为内容没有占满开辟的200字节大小的缓冲区，后面为乱码

### 读取buffer

buffer.toString()

### 拷贝buffer

sourceBuffer.copy(targetBuffer)

## fs文件基本操作

file system文件系统模块，简称fs，主要用于操作文件（读取，增加，修改，删除操作）。

### 引入文件模块：

由于fs是系统模块，所以直接引入。引入后就可以使用fs提供的方法。

var fs = require(‘fs’);

### 读取文件的方法：

#### fs.readFile(‘path’, options,callback) 异步读取

var fs = require(‘fs’);

fs.readFile(‘text1.txt’, function (err,data){...})

err:发生错误，抛出错误对象

data：读取到的数据

fs.readFile(‘text1.txt’, function (err,data){

if(err){

console.log(‘读取错误，原因：’+err.message);

//或者： throw err;但是不友好

}else{

console.log(‘读取到的文件数据是：’+data);

}

})；

console.log(‘因为是异步，所以这句话其实先执行’);

#### fs.readFileSync(‘path’) 同步读取

var fs = require(‘fs’);

console.log(‘同步开始：’);

var result = fs.readFileSync(‘0201.text’);

console.log(‘读取到的同步文件数据是：’+result);

console.log(‘因为是同步，依次执行’);

如果后面的模块依赖前面的模块，就是用同步读取；否则用异步。

### 写入文件的方法：

#### fs.writeFile(‘path’,’content’,callbcak) 异步写入

var fs = require(‘fs’);

var content = ‘我是写入的内容！’; //原内容会被覆盖掉

fs.writeFile(‘text1.txt’, content ,function (err){ //如果文件不存在直接创建

if(err){

console.log(‘写入错误，原因：’+err.message);

//或者： throw err;但是不友好

}else{

console.log(‘写入成功’);

}

});

//如果想要想要保留原来的内容

fs. readFile(‘text1.txt’, content ,function (err,data){

if(err){

console.log(‘读取错误，原因：’+err.message);

//或者： throw err;但是不友好

}else{

var oldCon = data; 所以干嘛要用变量保存啊

fs.writeFile(‘text1.txt’, oldCon +’新增内容不会覆盖前面内容’,function(err){

if(err){

console.log(‘追加错误，原因：’+err.message);

//或者： throw err;但是不友好

}else{

console.log(‘追加成功’);

}

})

}

});

### 文件信息fs.stat(path,callback)

var fs = require(‘fs’);

fs.stat(‘./text1.txt’,function(err,stats){

if(err){

console.log(‘状态获取失败，原因：’+err.message);

//或者： throw err;但是不友好

}else{

console.log(stats);

}

});

stats(status)的重要属性有：

stats.size:文件大小

stats.mtime:修改时间

（atime: access time访问时间，mtime:modify time文件被修改时间，ctime:change time文件的权限，拥有者，所属组，链接数发生改变的时间）

stats的重要方法有：

stats.isFile():是否是一个文件

stats.isDirectory():是否是一个文件夹

### 删除目录fs.rmdir(‘path’, callback)

fs.rmdir(‘./test.css’, function(err){

if(err){

console.log(‘删除发生错误！’);

}else{

console.log(‘删除成功！’);

}

});

### 删除文件fs.unlink(‘path’, callback)

fs.unlink(‘./test.css’, function(err){

if(err){

console.log(‘删除发生错误！’);

}else{

console.log(‘删除成功！’);

}

});

### 创建目录fs.mkdir(‘dirName’,callback)

应用场景:

var dirName = new Date().toaLocaleDateString(); //以每天的日期创建文件夹

fs.mkdir(dirName, function(err) {

if(err){

console.log(‘创建文件夹失败’+err.message); //已经存在的目录会报错

}else{

console.log(‘创建文件夹成功’);

}

});

### 获取目录中的文件或者目录

fs.readdir(path,options,callback)

fs.readdir(‘./styles’,function (err,filesArr){

if(err){

console.log(‘读取失败’+err.message);

}else{

console.log(‘读取的结果以数组的方式展示’);

for(var i in filesArr){

console.log((parseInt(x)+1)+”--”+fileArr[i]);

}

}

});

## Stream

应用程序中，流是一组有序的、有起点和终点的字节数据的传输方式。

传统的做法是把文件全部读取到缓冲区再操作，由于缓冲区使用内存，当文档很大就有可能造成内存溢出，所以要用流来操作。

### stream流的读取。

流还是fs文件操作模块。

fs.createReadStream(‘path’);

crs.setEncoding(‘设置流的编码方式’);

### 流的读取事件监听：

crs.on(事件类型)

data事件：默认每次读取64k

ends事件：

error事件：

### 流的写入

fs.createWriteStream(‘path’); //文件不存在的时候，自动创建

cws.writ

### 流的写入事件监听：

cws.on(事件类型)

finish事件：

error事件：

### 管道方式

pipe提供了一个输出流到输入流的机制，一般可用于大文件的复制。

语法：

createReadStream.pipe(createWriteStream)

### 链式流

链式：

通过连接输出流到另一个流并且多个对个流操作链的机制。链式流一般用于管道操作.

输出流.pipe（中转流）.pipe（输出流）

利用链式流，

压缩：

var zip = require(‘zlib’);

var pipezip = zip.createGzip();

crs.pipe(pipezip).pipe(cws);

## path模块

利用path提供的方法处理和转化文件的路径。

### path的方法：

path.normalize(p) 规范化路径

path.join([p1,p2.....]) 用于连接路径

path.dirname(p) 文件的名称

path.basename(p[,ext]) 文件名称

path.extname(p) 文件的扩展名

path.parse() 返回一个对象包含路径的信息

.dir 返回路径中的目录路径

.base 返回含扩展名的文件名

.ext 返回扩展名

.name 返回文件名

举例：/home/user/dir/file.txt

path.parse(‘/home/user/dir/file.txt’);

返回对象

{

‘root’: ‘/’,

‘dir’: ‘/home/user/dir’,

‘base’: ‘file.txt’,

’ext’: ‘.txt’, //包含.

‘name’: ‘file’

}

## URL模块

uniform resource locator统一资源定位符是对可以从，简称网址。

属性：

protocol: ‘https:’

slashes: true,

auth:null,

hostname: 主机名，包含端口

hash：#后面的值

host: 主机名

search：‘?....’查询参数

query：网址传参

pathname：目录路径

path：完整路径（含文件名）

href:完整网址

举例：

[https://search.jd.com:80/Search?keyword=xiaomi](https://search.jd.com/Search?keyword=xiaomi)

var url = require(‘url’);

var myURL = url.parse(‘https://search.jd.com/Search?keyword=xiaomi’);

返回对象

{

‘protocol’: ‘https:’,

‘slashes: true, //斜杠

‘auth: null, //认证(用户名和密码的验证)

‘host’: ‘search.jd.com:80’, //包含端口号

‘port: ‘80’,

‘hostname’: ‘search.jd.com’,

‘hash’: null, //#后面的参数

‘search’: ‘?keyword=xiaomi’, //搜索参数

‘query’: ‘keyword=xiaomi’, //网络传参

‘pathname’: ‘/item/url/110640’ //路径（目录）

‘path’: ‘/item/url/110640?fr=aladdin’ // 网址路径（包含文件名）

‘herf’: ‘https://baike.baidu.com:80/item/url/110640?fr=aladdin’ // 完整网址

}

# 2018.03.22

## 网络入门

### 主要的协议:

#### HTTP

http: hyper-text transfer protocol https:hyper-text transfer protocol saft

http是互联网上应用最为广泛的一种网络协议。

客户端--》request---》服务器

服务器--》response--》客户端

#### socket

socket本质是编程的接口（api）对TCP/IP的封装。TCP/IP也要提供可供程序员做网络开发所用的接口，这就是socket编程接口；HTTP是轿车，提供了封装或者显示数据的具体形式；socket是发动机，提供了网络通信能力。

---》keep online---》

客户端 服务器

《-- keep online--《

http与socket的区别：<https://www.cnblogs.com/meier1205/p/5971313.html>

## web客户端和web服务器

C/S：Client/Server 客户端/服务端 必须安装在客户端的软件，需要手动下载和更新（APP）

B/S: Browser/Server 浏览器/服务器模式 直接访问服务器端的软件（OA，CRM系统）

## HTTP

### http交互特点：

一次请求一次响应。

### http的状态码：

(1.消息（请求被接受需要继续处理）2.成功 3.重定向 4.请求错误（客服端） 5.6服务器错误)

200: ok

301: moved （资源被永久转移到其他url）

304: not modified

404: not found

500: internal server error

### MIME

(Multipurpose Internet Mail Extensions)多用途网络邮件扩充协议

txt text/plain(head默认)

html text/html

json text/json

jpg image/jpg

png image/png

gif image/gif

svg image/svg

doc application/msword

mps audio/mpeg

mpeg video/mpeg

### http模块

http.get(path,function(){..})

### 爬虫

网络爬虫（网络蜘蛛，网络机器人）是一种按照一定规则，自动地抓取万维网信息的程序或脚本。

搜索引擎search engine就是一个大的爬虫程序，比如：baiduspider google robots.txt

爬虫的实现原理：

通过http.get()请求获取网页的内容，根据内容的时间决定此内容是否是新内容，如果是新的内容就写入自己的索引数据库，以后搜索者查询关键词时，从数据库中执行select...where keywords=’ss’最后响应到客户端。

实现方式：

分析网页的结构，确定需要使用的资源。

通过正则表达式匹配到需要的内容

修饰符 i g m(多行)

元字符 .(除了换行和结束，任意单个字符)

量词 +（至少一个）\*（0-多）

返回匹配的结果 exec

发起http.get()请求，通过管道流的方式把网上资源写到本地。

## 浏览器客户端提交数据的方式

1. get提交

会在url地址栏后面附上表单的关键词和值

https://search.jd.com/Search?keyword=mate10

1. post提交

只有表单的method=post这种方式才能做post提交，安全性高，传输的数据量大  
因为POST方式传值是以请求体的方式流到服务器的，所以必须注册 req请求对象的data事件来接收。

1. 超链接的参数，跟get方式类似

http://e.cn.miaozhen.com/r/k=2022917&p=72JaP&ro=sm&dx=0

## 模板网页的使用步骤

1. 提供模板内容html文件

2. 提供一个json数据

3. 模板语法：

<%if(true){%>

<%=value变化%>

<%}%>

4. 安装ejs模块

npm i ejs --save

5. 引入ejs模板

6. 使用ejs模板提供的方法渲染模块文件

ejs.renderFile("模板文件","数据源",callback(err,渲染后得到的模板和数据的合成体){});

# 2018.03.23

## Express

### 什么是express

是对node.js底层的功能进行封装一个node.js服务器框架。

### express核心

中间件来响应HTTP请求

路由来响应不同的HTTP请求动作

url资源映射

模板动态渲染HTML页面

### 如何使用express

express-generator脚手架工具

全局安装 cnmp i -g express-generator

创建项目 -e filename使用ejs模板 express -e filename

change directory:

> cd project

install dependencies:

> npm install

run the app:

> SET DEBUG=project:\* & npm start 【project是项目文档名filename】

### express文件夹详解

1. bin/www命令文件夹（一般不用，为Linux准备）

npm start本质是执行bin/www

“script” : {

“start”:”node ./bin/www”

}

1. node\_modules第三方模块的安装文件夹
2. public 公共的资源文件夹，可以直接访问，但不用加public路径，优先级高于路由。

支持图片，脚本，样式，HTML等等。。。

<http://127.0.0.1:3000>/ ---> <http://127.0.0.1:3000>/public/index.html

总结：静态网页全部放进public目录

1. routes路由文件夹

所有的路由文件都放到此目录，路由文件就是响应http请求文件，相当于MVC的控制器。

MVP: model view controller是模型-视图-控制器的缩写，一种软件设计典范，用一种业务逻辑，数据，界面显示分离的方法组织代码，将业务逻辑聚集在一个部件里，在改进和个性化定制页面及用户交互的同时，不需要重新编写页面业务逻辑。

1. views 视图文件夹，存放模板文件的地方，就是MVC视图。由于使用ejs模板，