Nodejs 知识总结

Node.js 概述

Node.js 是 JS 运行在后端的环境，使用这个环境可以进行后端开发，实现 Java，Python 等后端语言同样的功能。

1.对比 JS

JS 运行在客户端浏览器，Node.js 运行在后端（服务器端）

JS 存在多款浏览器有代码兼容性问题，Node.js 只有一种环境不存在代码兼容性问题

JS 和 Node.js 都有相同的自定义对象和内置对象，不同的宿主对象

JS 用于开发浏览器端的交互效果，Node.js 用于服务器端开发，例如：数据库访问，调用其它服务器..

2.下载安装

www.nodejs.org

3.运行 JS

脚本模式

node 拖拽文件 回车

交互模式

node 回车 进入交互模式

连续两次 ctrl+c 或者一次 ctrl+d 退出交互模式

全局对象——global

全局对象：可以在任意作用域的使用的对象

可以用来检测一个变量或者函数是否为全局的

练习：编写脚本文件 06_global.js，声明变量，创建函数；使用 global 检测是否为全局的。

浏览器下的 global 改名字为 window

练习：编写 07.js 和 07.html，把 js 嵌入到 html 中；

在 js 声明变量，创建函数；使用 window 检测是否为全局的

浏览器：每个 js 文件下是在全局作用域，存在全局污染 Node.js：每个 js 文件都是在一个独立的文件作用域，不存在全局污染

模块

在 Node.js 下每一个文件就是一个模块

项目中需要哪些文件，都需要把文件引入进来

require() 用于引入其它的模块，同一级路径的文件需要加./

例如：require('./08_yan.js')

module.exports 当前模块暴露的对象，默认是一个空对象，如果要暴露哪些内容，需要添加到这个对象下

模块分类

分为自定义模块、核心模块、第三方模块

	以路径开头	不以路径开头
文件模块	require('./circle.js') 用于引入自定义模块	
目录模块	require('./03_tao') 会到目录下寻找 package.json 中 main 属性对应的文件，如果找不到会去查找 index.js	require('tao') 会到当前的目录下的 node_modules 目录中寻找 tao 模块，如果找不到会一直往上一级的 node_modules 目录寻找；常用于引入第三方模块

json 文件，是一种存数据的文件，只能放数组或者对象；对象的属性名必须加双引号，值是字符串必须用双引号

{ "a": "tao" }

包和 npm

CommonJS：是一套模块化规范，Node.js 的引入模块和暴露对象都是基于这个规范，Node.js 就是使用的这套模块化规范。

包：package，指的是第三方模块

npm：用来管理包的工具，例如：下载、上传、升级、卸载... 在 Node.js 安装的时候就已经附带安装了。 npm -v

www.npmjs.com

1.npm 命令

npm init 一直回车，用于创建 package.json 文件，作为项目描述文件

npm install 模块名称 下载安装指定的模块，会将模块下载再 node_modules 目录下，如果目录不存在会自动创建。会下载其它依赖的包。生成一个文件 package-lock.json，用于记录所有包的信息。

npm install 会自动的去下载 package-lock.json 和 package.json 中记录的包

其它所有 npm 命令： www.npmjs.cn

nodejs 中文文档： www.nodejs.cn

核心模块——查询字符串模块(querystring)

查询字符串：浏览器端向服务器端传递参数的一种形式，位于网址中

http://www.codeboy.com:9999/products.html?kw=小米&a=1

查询字符串模块：是 Node.js 下专门用于操作查询字符串的工具，可以获取传递的参数

parse() 将查询字符串转为对象，可以获取传递的值

练习：获取以下查询字符串的值 'user=tao&pwd=123456'，最后打印以下格式

登录成功！用户名：xxx 密码：xxx

网址模块(URL)

网址(URL)：统一资源定位，互联网上的任何资源都有对应的网址；最终通过网址获取服务器端的资源

http://www.codeboy.com:9999/product_details.html?lid=1

协议 域名/IP 地址 端口 请求的服务器端资源 查询字符串

new URL(网址) 将一个网址转为对象，目的获取各个部分

全局模块

也称为全局对象，不需要引入就可以直接使用的模块

1.console 模块

提供了一组用于控制台输出的方法

console.log(1) //打印日志

console.info(2) //打印消息

console.warn(3) //打印警告

console.error(4) //打印错误

console.time() //开始计时

console.timeEnd() //结束计时

注意事项：开始计时和结束计时的参数要保持一致。

2.process 模块(了解)

进程：系统上的每个软件运行都是代表一个进程，进程占用一定的 CPU、内存

process.arch 查看当前 CPU 架构

process.platform 查看当前的操作系统

process.pid 查看当前进程的编号，随机生成

process.kill()　用于结束指定编号的进程

3.Buffer 模块

缓冲区：是内存中的一块区域，用于临时存储数据

Buffer.alloc(6, 'abc 涛')　创建 Buffer，分配大小为 6 个字节，每个汉字占 3 个字节

toString()　将 Buffer 数据转为字符串，前提存储的是字符串。

4.timer 模块(定时器模块)

提供了一组全局函数

(1)一次性定时器

开启 var timer=setTimeout(回调函数，间隔时间) 当间隔时间到了，调用一次回调函数；间隔时间单位是毫秒
清除 clearTimeout(timer)

(2)周期性定时器

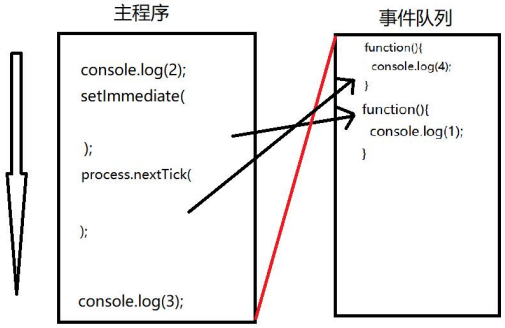
var timer=setInterval(回调函数，间隔时间) 每隔一段时间调用一次回调函数
清除 clearInterval(timer)

(3)立即执行定时器

setImmediate(回调函数)/clearImmediate()
process.nextTick(回调函数)

定时器：所有的定时器都会进入到事件队列执行；

事件队列：由一组排队执行的回调函数组成，只有执行完主程序才会执行事件队列



同步和异步

同步：在主程序执行，同步的方法通过返回值获取结果。

异步：会产生回调函数，并放入到事件队列执行；只有主程序执行完才会执行事件队列；　所有的定时器函数都是异步的，Node.js 下提供的带有回调函数的都是异步的；Node.js 下提供的异步方法在一个独立的线程执行，通过回调函数获取结果。

文件系统模块(fs)

用于文件的操作

文件分为目录形式和文件形式

使用这个模块，必须先引入

```
const fs=require('fs')
```

(1)查看文件的状态

statSync(文件的路径)/stat(文件的路径，回调函数)

isDirectory()　是否为目录

isFile()　是否为文件

(2)创建目录

mkdirSync(目录的路径)/mkdir(目录的路径，回调函数)

(3)移除目录

rmdirSync(目录的路径)/rmdir(目录的路径，回调函数)

只能移除空目录

1.读取目录

readdir(目录的路径，回调函数)/readdirSync(目录的路径)

读取的结果是数组

练习：使用同步方法读取目录 ../day03

2.清空写入文件

writeFile(文件的路径，写入的数据，回调函数)/writeFileSync(文件的路径，写入的数据)

如果文件不存在先创建文件，然后写入数据

如果文件已经存在先清空文件内容，然后写入数据

练习：使用同步方法往文件 2.txt 中写入数据

3.追加写入文件

appendFile(文件的路径，写入的数据，回调函数)/appendFileSync(文件的路径，写入的数据)

如果文件不存在先创建文件，然后写入数据

如果文件已经存在，在文件的末尾追加写入数据

练习：使用同步方法往文件 3.txt 中写入数据

4.读取文件

readFile(文件的路径，回调函数)/readFileSync(文件的路径)

练习：使用异步方法读取 data.txt 文件的数据

练习：使用同步方法读取 1.txt 文件的数据

5.删除文件

unlink(文件的路径，回调函数)/unlinkSync(文件的路径)

练习：使用异步方法删除 1.txt；使用同步方法删除 2.txt

6.检测文件(目录)是否存在

existsSync(文件路径)

7.拷贝文件

copyFile(源文件,目标文件,回调函数)/copyFileSync(源文件,目标文件)

练习：将文件 data.txt 拷贝到当前目录下名称为 data1.txt

文件流(stream)

可以把文件分成很多段

createReadStream()　创建可以读取的文件流

createWriteStream()　创建可以写入的文件流

pipe()　管道，可以将读取的流添加到写入的流，最终完成文件拷贝

on(事件名称，回调函数)　添加事件，用于监听某一个操作，一旦监听到会自动调用回调函数；　事件名称是固定的字符串形式。

HTTP 协议

WEB 服务器：为浏览器端提供资源的服务器，例如：网页，图片，数据...

HTTP 协议：超文本传输协议，是浏览器端和 WEB 服务器之间的通信协议

1.通用头信息

既包含一部分请求的也包含一部分响应

Request URL：要请求的资源

Request Method：请求的方法，对资源的操作方式，　get/post...

Status Code：响应的状态码

1**：接收到了请求，还没有结束响应

2**：成功的响应

3**：响应的重定向，会跳转到其它的资源

4**：客户端错误

5**：服务器端错误

2.响应头信息(Response)

服务器端发出的

Location：设置要跳转的 URL，通常结合着状态码 302 使用

Content-Type：响应的内容类型，解决中文乱码 'text/html; charset=utf-8'

3.请求头信息(Request)

浏览器发出的

http 模块

使用 http 模块可以创建 WEB 服务器，为浏览器端提供服务

1.创建服务器

```
const http=require('http');
const app=http.createServer();//创建 WEB 服务器
app.listen(3000)    //设置端口为 3000
```

2.接收请求，做出响应

```
//通过事件监听浏览器的请求
app.on('request', (req,res)=>{
  res 响应的对象
  res.setHeader()  设置响应的头信息
  res.wirte()  设置响应到浏览器的内容
  res.end()  结束并发送响应
})
```

res.statusCode=值 设置响应的状态码

req 请求的对象

req.url 获取请求的服务器端资源 是字符串形式 '/xxx'

req.method 请求的方法

框架：是一整套解决方案，简化了已有功能，添加了之前没有的功能，专门用于项目开发。任何的语言开发项目都会选择使用框架。

Nodejs: express koa egg

前端：Vue React Angular

Java：Spring....

express 框架

基于 Node.js 平台，快速、开放、极简的 WEB 开发框架

属于是第三方模块，需要先去下载安装

npm install express

1.创建 WEB 服务器

```
const express=require('express'); //引入下载的模块
const app=express(); //创建 WEB 服务器
app.listen(3000);  //设置端口
```

2.路由

路由用来监听不同的请求，路由包含三部分：请求的方法、请求的 URL、回调函数

(1)响应的对象

res.send() 设置响应的内容并发送

res.redirect() 设置响应的重定向并发送

res.sendFile() 设置响应的文件并发送，使用绝对路径

__dirname+'/文件名称'

```
__dirname  获取当前模块的绝对路径
```

(2)请求的对象

req.method　获取请求的方法

req.url　获取请求的 URL

req.query　获取 get 传递的参数，并转为对象

req.params　获取路由传递的参数，并转为对象

传参方式	格式	路由获取
get 传递	http://127.0.0.1:3000/mysearch? kw=笔记本	req.query {kw: '笔记本'}
路由传参	http://127.0.0.1:3000/detail/10	①先指定参数名称 ②req.params {id: 10}

post 传参

相对安全，不会将数据暴露在 URL

没有大小限制，可以传递字符串，也可以传递大文件(视频,声音...)

现阶段发送 post 请求，只能采用表单提交

```
在路由之前设置，将 post 传递的参数转为对象
app.use( express.urlencoded({
   extended: true   //内部如何转为的对象，是否使用第三方模块
}) )
```

```
在路由获取
req.body
```

路由器

用于将所有的路由按照不同的功能模块进行分类，目的是便于管理路由。

```
1.路由器
//创建路由器对象
const r=express.Router();
//添加路由
//暴露路由器对象
module.exports=r;
```

```
2.WEB 服务器
//引入路由器
//使用路由器，就会将所有路由器的路由挂载到 WEB 服务器;同时给路由添加前缀
app.use( 添加的前缀，引入的路由器 )
```

练习：添加商品路由器 product.js，创建路由器对象，添加删除商品路由(get　/delete)，暴露路由器对象；最后在 app.js 下引入路由器，使用路由器并添加前缀/product

中间件

中间件是用于拦截对 WEB 服务器的请求

express 下中间件分为应用级中间件、路由级中间件、内置中间件、第三方中间件、错误处理中间件

1.应用级中间件

也称为自定义中间件，就是一个函数，一旦拦截到会自动调用这个函数。

```
function   fn(req,res,next){
   next(); //往后执行下一个中间件或者路由
}
app.use( 拦截的 URL, fn )
```

2.路由级中间件

路由器的使用就是

app.use(拦截的 URL，路由器)

3.内置中间件

指的是 express 提供的中间件，可以直接使用的

(1)将 post 传递参数转为对象

app.use(express.urlencoded())

不写要拦截的 URL，表示拦截所有的 post 请求

(2)托管静态资源

静态资源：指的是固定的文件，例如：html、css、js、图像....

托管静态资源，如果浏览器要请求静态资源，不需要通过路由响应，而是让浏览器自动去到某个目录下寻找文件。

app.use(express.static('要托管的目录'))

mysql 模块

是 Node.js 下一个专门用于操作 mysql 数据的模块

属于是第三方模块，需要先去下载安装

npm install mysql

createConnection() 创建普通连接

createPool() 创建连接对象，产生一组连接

query(SQL 命令，数组，回调函数) 执行 SQL 命令，通过回调函数获取结果，数组用于对数据进行过滤，过滤后再去替换占位符(?)

mysql 模块下解决 SQL 注入：使用占位符，对所有用户提供的值进行过滤，过滤后不再破话原有的 SQL 命令。

数据接口(RESTful 规范接口)

后端为前端提供的动态资源

每次写的一个路由就是一个接口

登录接口

接口地址：http://127.0.0.1:3000/mylogin

请求方式：post

传递的参数：用户名/密码

返回结果：登录成功，欢迎 xxx

1.接口地址

代表的是要请求的资源

http://127.0.0.1:3000/v1/users

版本号 资源名称(复数)

http://127.0.0.1:3000/v1/users/login

对资源特殊操作

2.请求方法

对应的是资源的操作方式——增删改查

post 插入数据

delete 删除数据

put 修改数据

get 查询数据

3.传参方式

操作单个资源，使用路由传参

http://127.0.0.1:3000/v1/users/2

添加和修改，使用 post 传参

从一组数据中进行过滤，例如分页，使用 get 传递

http://127.0.0.1:3000/v1/users?pno=1&count=10

过滤出工资在 6000~8000

http://127.0.0.1:3000/v1/emps?s1=6000&s2=8000

4.返回结果

json ——— 字符串形式的对象，属性名必须用双引号，属性值是字符串的话必须用双引号，包含一个人为规定状态码

{"code":200, "msg": "登录成功"}

{"code":500, "msg": "登录失败"}

{"code":200, "msg":"获取成功","data": 查询到的数据}

学子商城项目回顾

1.在 app.js 下创建 WEB 服务器

2.在用户路由器 user.js 下，创建路由器对象，添加用户注册的路由(post /reg)，将路由器对象暴露出来

3.在 app.js 中引入用户路由器(./routes/user.js)，使用路由器，添加前缀/v1/users

/v1/users/reg

4.用 apipost 测试接口

5.获取 post 传递的参数

6.在 pool.js 中，创建连接池对象，暴露这个对象。

7.在 user.js 中引入连接模块(../pool.js)，执行 SQL 命令，将数据插入到数据表 xz_user

错误中间件

在所有路由(器)后边

```
app.use( (err, req, res, next)=>{
    err 接收的传递过来的错误
    res.send({code:502, msg: '服务器端错误' });
} )
```

正则表达式

1 8 233489834

1 3~9 连续 9 个 0~9 数字

字符串格式规则

[] 字符集，可以作为备选方案

- 连续范围简化

常用的正则表达式

https://www.cnblogs.com/hsinfo/p/13584432.html

正则表达式学习视频

https://pan.baidu.com/s/1MvzutFpYkYmqj9CdKkn4-g 提取码：hsdq

Cannot set headers after they are sent to the client

有多个 send()被调用了；

一个 send 调用表示响应结束了，不允许多次响应