目 录

[day01 1](#_Toc10097551)

[1.ES6 1](#_Toc10097552)

[2.nodejs概述 1](#_Toc10097553)

[1.全局对象 1](#_Toc10097554)

[2.模块 2](#_Toc10097555)

[day02 3](#_Toc10097556)

[1.模块系统 3](#_Toc10097557)

[2.包和npm 3](#_Toc10097558)

[3.querystring模块 4](#_Toc10097559)

[4.url模块 4](#_Toc10097560)

[5.fs模块——文件系统模块 4](#_Toc10097561)

[day03 5](#_Toc10097562)

[1.fs模块 5](#_Toc10097563)

[2.http协议 5](#_Toc10097564)

[3.http模块 6](#_Toc10097565)

[4.express框架 6](#_Toc10097566)

[day04 7](#_Toc10097567)

[1.使用zlib模块对服务器端响应压缩 8](#_Toc10097568)

[2.express 8](#_Toc10097569)

[3.路由器 9](#_Toc10097570)

[4.中间件 10](#_Toc10097571)

[day05 11](#_Toc10097572)

[1.中间件 11](#_Toc10097573)

[2.mysql模块 12](#_Toc10097574)

[day06 13](#_Toc10097575)

# day01

## 1.ES6

(1)函数中的参数

如果没有为形参设置默认值，如果没有传递实参，则自动调用形参的默认值

function add(a,b,c=0){ };add(2,5,8);//15

(2)模板字符串

`在模板字符串中可以写任何形式的字符串${js表达式}`

练习：创建一个图书对象，包含有图书的编号，名称，价格，上架时间，是否在售(1->是/0->否)，使用模板字符串打印书的内容

## 2.nodejs概述

基于谷歌的V8引擎，运行在服务器端的语言。

对比JS和NodeJS

1. JS运行在浏览器端，存在多个浏览器，容易产生兼容性的问题；而NodeJS运行在服务器端，只有V8引擎一种解释器，不存在兼容性问题。
2. 两者都有内置对象、自定义对象，宿主对象(根据执行环境的不同)
3. JS用于浏览器中的交互效果，而NodeJS用于服务器端，例如web服务器创建，数据库操作，文件操作。

官网 nodejs.org

中文版 nodejs.cn

NodeJS执行方式：

脚本模式：node c:/xampp/...1.js 回车

交互模式：node 回车进入交互模式

退出交互模式，两次Ctrl+c 或者 .exit

### 1.全局对象

NodeJS:global

在交互模式，声明的变量和创建的函数默认都属于全局对象下的，可以使用global来访问。var a=1;global.a

练习：在脚本模式下，使用global来访问变量和函数

在脚本模式，声明的变量和创建的函数默认都是局部作用域下，不能被global访问，好处在于不能污染全局。

JS:window

在浏览器下，脚本中声明的变量和创建函数默认都是全局作用下的，可以使用window来访问，var a=1;window.a

(1)console对象

console是global下的成员，可以使用global.console访问

global.console.log() 打印日志

console.info() 打印消息

console.warn() 打印警告

console.error() 打印错误

console.time('自定义字符串') 开始计时

console.timeEnd('自定义字符串') 结束计时

开始计时和结束计时中的字符串要保持一致

练习：查看for/while/do-while循环相同次数的耗时情况。

(2)process对象

进程：当操作系统启动一个程序时，会将必须的可执行文件和数据文件从系统调入到内存，分配到必要的内存空间。

对应操作系统中的任务管理器

process.arch 查看当前CPU架构

process.platform 查看当前的操作系统

process.env 查看当前操作系统的环境变量

process.version 查看当前NodeJS版本

process.pid 查看当前进程的编号

process.kill() 结束某个编号的进程

(3)Buffer对象

缓冲区：在内存中存储数据的区域，常用于存储网络传输时的资源

创建Buffer:

Buffer.alloc(5,'abcde');

将Buffer数据转为字符串:

buf.toString()

(4)全局函数

parseInt/parseFloat...

①一次性定时器

开启:

var timer=setTimeout(回调函数，间隔的时间);

当间隔的时间到了，执行一次回调函数，单位是毫秒

清除:

clearTimeout(timer);

②周期性定时器

开启:

var timer=setInterval(回调函数，间隔的时间);

当间隔的时间到了，执行一次回调函数，单位是毫秒

清除:

clearInterval(timer);

③立即执行

开启:

var timer=setImmdiate(回调函数);

回调函数在事件队列中执行，在事件的最前面执行

清除:

clearImmdiate(timer);

④立即执行

process.nextTick(回调函数)

在事件的最后执行

### 2.模块

模块是一个独立的功能体

在NodeJS下模块分为3类：自定义模块、核心模块、第三方模块

自定义模块:

在NodeJS下，每一个文件都是模块

NodeJS自动为每个文件添加了构造函数，所有的代码被构造函数所包含

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

程序员写的代码

})

require('08\_2.js'); //用于引入一个模块

module:当前的模块对象

module.exports:当前模块的导出对象，可以提供其他模块使用的属性和方法

课后任务：

(1)复习，整理思维导图

(2)练习：创建两个模块main.js(主模块),circle.js(功能模块);在功能模块下

创建两个函数，传递1个参数(半径),分别获取圆的面积和周长，导出这两个函数; 在主模块中引入功能模块，并调用两个方法

(3)预习模块querystring(查询字符串),url,fs

# day02

复习

nodejs

global

console.log/info/warn/error/time/timeEnd

process.arch/platformersion/pid/kill()

var buf=Buffer.alloc(5,'abcde') buf.toString() String(buf)

setTimeouttIntervaltImmediate/process.nextTick

## 1.模块系统

每个文件就是一个模块

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

var a=1;

})

require 引入另一个模块

module 当前模块对象

module.exports 导出(暴露)的对象

exports 指代module.exports

\_\_filename 当前模块的完整路径和文件名称

\_\_dirname 当前模块的完整路径

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’)常用于用户自定义的模块，如果后缀名为js,则可以省略 | Require(‘url’)常用于引入官方提供的核心模块 |
| 目录模块 | Require(‘./02\_2’)到02\_2目录下寻找package.json文件main属性对应的模块，如果找不到会去引入index.js | Require(‘04\_2’)会自动到当前目录下的node\_modules中寻找目录，如果找不到则会到上一级知道顶级目录，常用于引入第三方模块 |

练习：创建模块03\_1.js，引入当前目录下03\_2目录模块，在03\_2目录有文件fun.js,导出一个函数add(计算两个数字相加);在03\_1.js中调用该函数。

练习：在05目录下创建05\_1.js,引入不以路径开头的目录05\_2,含有文件foucs.js，在文件中导出函数，在05\_1.js调用该函数。

## 2.包和npm

CommonJS是一个标准贵干，是JS向服务器端发展的规范

包(package)就是一个目录模块

npm:是用于管理包的工具

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

**切换到安装的目录**

cd 完整路径

进入要安装的目录，在空白区域按住shift键，单击右键，选择在此处打开powershell窗口

**使用npm安装**

npm install 包的名称 回车

自动安装在当前目录下的node\_modules中

npm init –y 初始化一个package.json文件，可以记录安装的包

npm install 自动安装package.json文件中记录的包

## 3.querystring模块

查询字符串：浏览器项服务器发请求，传递数据的一种方式

<https://search.jd.com/Search?keyword=apple&lid=1>

parse() 将查询字符串格式化为对象，获取数据

stringify() 将对象转换为查询字符串

练习：获取查询字符串中的关键字pug(rsv\_idx=1&tn=baidu&wd=pug);

## 4.url模块

parse() 将url格式化为对象

protocol 协议

hostname 域名/ip地址

port 端口号

pathname 请求的文件在服务器上的路径

query 查询字符串

format() 将对象转换成查询字符串

练习：获取url中查询字符串中的数据

(<http://www.tmooc.cn:80/web/1904.html?sid=10&sex=1>)

步骤：将url格式化为对象，获取到查询字符串；在获取的查询字符串格式化为对象，获取数据08.js

## 5.fs模块——文件系统模块

(1)查看文件的状态

fs.stat(fd,callback)

fd 文件的路径

callback 回调函数，用来获取结果

err 产生的错误信息

stats 具体文件状态信息

isDirectory() 是否为目录

isFile() 是否为文件

(2)进程和线程

进程是操作系统分配的占用内存的基本单位，一个进程可以有一个或者多个线程。

nodejs运行程序的时候，由一个线程来完成。

(3)同步和异步

同步：会阻止后续代码执行，只有前边的操作全部结束，才能开始后续的操作，是通过返回值来获取结果。

异步：不会阻止后续代码执行，后边的操作可以先执行，不必等待前边的操作结束，是通过回调函数来获取结果。

(4)创建目录

|  |
| --- |
| mkdir(fd,callback)/mkdirSync(fd)  fd 表示目录的路径 |

(5)移除目录

|  |
| --- |
| rmdir(fd,callback)/rmdirSync(fd) |

练习：使用同步和异步移除mydir和mydir2

(6)读取目录

|  |
| --- |
| readdir(fd,callback)/readdirSync(fd) |

课后任务：

1. 复习，整理思维导图
2. 练习：在homework目录下创建模块main.js,引入不带路径的目录模块hello,含有

文件my.js,在文件中导出一个函数，可以创建目录。在main.js中调用该函数。

1. http协议和http模块

# day03

## 1.fs模块

(1)创建文件/写入文件

|  |
| --- |
| writeFile(fd,data,callback)/writeFileDync(fd,data)  fd 文件的路径  data要写入的数据  callback 回调函数  如果文件不存在会创建并写入，如果文件已经存在，清空里边内容然后写入 |

(2)追加写入

|  |
| --- |
| appendFile(fd,data,callback) |

练习：使用追加写入数据abcde到文件2.txt,运行多次。

练习：将一组数据写入到文件3.txt[‘biubiu’,’平凡而伟大’,’到南方去’,’到北方去’];

(3)读取文件

|  |
| --- |
| readFile(fd,callback)/readFileSync(fd)  读取的结果是buffer数据，需要转为字符串 |

(4)删除文件

|  |
| --- |
| unlink(fd,callback)/unlinkSync(fd) |

(5)判断文件/目录是否存在

|  |
| --- |
| existsSync(fd) 存在->true 不存在->false |

(6)文件流Stream

|  |
| --- |
| var readStream=createReadStream(文件路径) 创建可读流  readStream.on(‘data’,function(chunk){  //当有数据进入到内存中，自动触发该事件  //分块来读取数据  chunk 就是读取某一块数据  })  readStream.on(‘end’,function(){ })  当数据读取结束，触发该事件  var writeStream=fs.createWriteStream(文件路径)  pipe 管道，可以将数据使用管道写入到文件  readStream.pipe(writeStream); |

## 2.http协议

是浏览器和web服务器之间的通信协议

(1)通用头信息 General

Request URL:请求的URL，对应的浏览器地址栏，向服务器获取哪些内容

Request Method:请求的方法，获取数据的方式post/get

Status Code:响应的状态码

1\*\*：正在响应，还没有结束

2\*\*：成功的响应

3\*\*：响应的重定向，会跳转到另一个网址

4\*\*：客户端请求错误

5\*\*：服务器端错误

Remove Adress：请求的服务器的ip地址和端口号

(2)响应头信息 Response headers

Connection：连接方式 keep-alive 持久连接

Content-type:响应的文件类型

Content-Encoding:压缩形式 gzip

Transfer-Encoding:传输方式(流Stream),chunked分块传输

Location:跳转的URL，常配合着状态码3\*\*使用。

(3)请求头信息 Request Headers

Accept:浏览器端接收的文件类型有哪些

Accept-Encoding:浏览器端接收的压缩形式

User-Agent:发送请求所使用的浏览器

(4)请求主体

可有可无，浏览器向服务器发请求传递的数据

## 3.http模块

既可以模拟浏览器向服务器发请求，还可以创建web服务器

(1)模拟浏览器

|  |
| --- |
| http.get(url,callback)  get 请求的方法  callback 回调函数，用于获取服务器端的响应  res 响应的对象  res.statusCode 获取响应的状态码  res.on(‘data’,function(chunk){ })  事件：监听服务器端是否有数据流传输进来  chunk 就是传输的数据，格式默认为buffer |

(2)创建web服务器

|  |
| --- |
| var app=http.createServer();//创建web服务器  app.listen(8080);//监听端口  app.on(‘request’,function(req,res){  //监听浏览器的请求  req 请求的对象  req.method 请求的方法  req.url 请求的URL  req.headers 请求的头信息  res 响应的对象  res.writeHead(statusCode,headers) 设置响应的状态码和头信息  res.write() 设置响应的内容  res.end() 结束并发送响应到浏览器端  }); |

练习：创建web服务器，监听端口8081，接收浏览器的请求，根据请求，做出不同的响应 (/login 响应内容 this is login page;/reg 响应内容 this is reg page;/study 跳转到 <http://www.tmoo.cn)以上都没有，响应内容> 404 not found

## 4.express框架

基于NodeJS,快速、开发、极简的web开发框架

<http://www.npmjs.com/package/express>

http://www.expressjs.com.cn

下载安装：nmp install express

|  |
| --- |
| const express=require(‘express’);//引入第三方的express  var app=express();//创建web服务器  app.listen(8080);//监听端口 |

课后任务：

1. 复习，整理思维导图
2. 练习：使用http模块，创建web服务器，接收浏览器的请求，根据请求做出不同的响应（/index 响应内容 this is homepage;/login 响应内容 <h2>please login</h2>;/ 跳转到 /index;其它 响应 404 not found）
3. 预习：express中的路由，get和post

# day04

复习

fs

mkdir/rmdir/readdir/create/writeFile/appendFile/readFile/unlink/existsSync/createReadStream/createWriteStream

http协议

通用头信息 请求方法，请求URL，响应状态码

请求头信息

响应头信息 响应文件类型，压缩类型 gzip

请求主体

http模块

http.get(url,function(res){

res.statusCode;

res.on(‘data’,function(chunk){ });

});

http.createServer();

listen(8080);

app.on(‘request’,function(req,res){

req.method/url/headers

res.writeHead 设置响应的头信息

res.write 设置相应的内容

res.end 结束信息

})

express

## 1.使用zlib模块对服务器端响应压缩

|  |
| --- |
| //1.引入zlib模块  const zlib=require(‘zlib’);  //2.设置内容压缩类型  ‘Content-Encoding’:’gzip’  //3.创建压缩  var gzip=zlib.createGzip()  //4.将读取的文件流放入到压缩，发送到浏览器  fs.createReadStream(‘demo.html’).pipe(gzip).pipe(res) |

## 2.express

（1）路由

浏览器向web服务器发请求，web服务器根据请求的方法和请求的URL来作出响应。

三要素：请求的方法、请求的URL、响应

res 响应对象

send() 响应一段内容

redirect() 设置响应的重定向，也就是跳转

sendFile() 响应文件，要求使用绝对路径 \_\_dirname

注意：在路由中响应只能使用一次

req 请求对象

method 获取请求的方法

url 获取请求的URL

headers 获取请求的头信息

query 获取get请求的数据（查询字符串），返回对象

练习：创建web服务器，添加多个路由(/login 响应’这是登录页面’；/study 响应跳转到[http://www.tmooc.cn;/reg 响应一个文件reg.html](http://www.tmooc.cn;/reg%20响应一个文件reg.html))

（2）路由传递数据

设置路由中接收的名称

|  |
| --- |
| app.get(‘/detail/:lid’,function(req,res){  req.params //获取传递的数据，格式为对象  }); |

浏览器传递数据

http://127.0.0.1:8080/detail/27

练习：创建购物车路由，请求方法:get,请求<url:/shopping>,传递商品的价格price和名称pname,接收传递的数据，并将数据响应到浏览器中

商品模块

列表 详情 修改 删除

/list /detail /update /delete

用户模块

列表 详情 修改 删除

/list /detail /update /delete

## 3.路由器

路由在使用的过程，不同的模块下的路由可能出现相同的URL，将不同模块下的路由区分开来，把同一模块下的路由放到同一个路由器。

创建路由器

|  |
| --- |
| const express=require(‘express’);  var router=express.Router();//创建空的路由器对象  router.get(‘/list’,function(req,res){ });//添加路由  module.exports=router;//导出路由器对象 |

Web服务器使用路由器

|  |
| --- |
| const productRouter=require(‘./product.js’);  app.use(‘/product’,productRouter);//把使用路由器，并在路由的url中添加前缀，访问形式/product/list |

练习：创建用户路由器user.js,添加用户列表，导出路由器；在web服务器下引入并使用，添加前缀/user

## 4.中间件

中间件为主要的业务逻辑服务

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，是一个函数，需要配合路由或者其他中间件一起使用。

app.user(‘/reg’,function(req,res,next){ })

拦截url为/reg的路由

app.user(function(req,res,next){ })

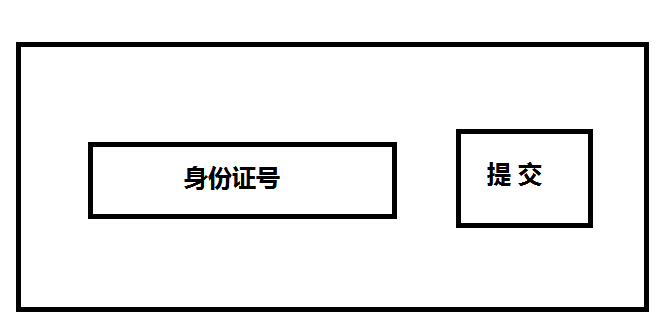
拦截所有的路由

next();执行下一个中间件或者路由。

课后任务：

（1）复习，整理思维导图

（2）练习：创建web服务器，向服务器请求查询生日的html文件,输入身份证号，点击提交，响应生日和性别到浏览器端



（3）复习mysql

# day05

## 1.中间件

（1）应用级中间件

（2）路由级中间件

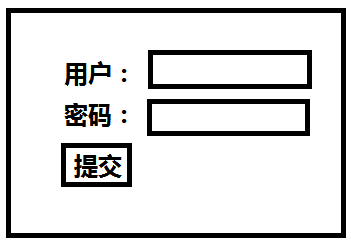
路由器的使用 app.use(‘use’,用户路由器)

（3）内置中间件

app.use( express.static(‘托管的目录’) );

把所有的静态资源托管到某个目录，如果浏览器请求文件，会自动到该目录下查找，不需要再创建单独路由响应文件。

练习：托管静态资源到public下，向服务器端请求login.html,点击提交，再次想服务器发请求post /losin 在服务器端获取提交的数据



（4）第三方中间件

body-parser中间件的使用

用于将post请求的数据格式化为对象

|  |
| --- |
| //1.下载安装并引入  const bodyParser=require(‘body-parser’);  //2.使用中间件  app.use(bodyParser.urlencoded({  extended:false //不使用第三方的qs模块格式化为对象，而是使用核心模块querystring  }));  //3.在post请求的路由中  req.body 返回一个对象 |

（5）错误中间件

## 2.mysql模块

mysql.exe –h127.0.0.1 –P3306 –uroot –p

INSER INTO emp VALUES(…)

DELETE FROM emp WHERE eid=5;

UPDATE emp SET sex=0,salary=8000 WHERE eid=6;

SELECT \* FROM emp;

|  |
| --- |
| var connection=mysql.createConnection({ })创建连接对象，在参数中传递连接mysql服务器的相关信息  connection.connect() 建立连接  connection.query(aql,callback) 执行SQL语句，参数sql就是要执行的SQL语句，callback回调函数用于获取SQL语句的结果  connection.end() 关闭连接 |

查询数据，返回数组

插入/修改/删除数据，返回对象，通过affectedRows可以判断是否执行成功，如果大于0成功。

SQL注入

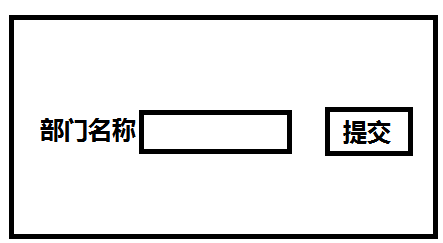
SELECT \* FROM user WHERE uname=’root’ and upwd=123456 **OR 1=1**;

使用连接池

|  |
| --- |
| var pool=mysql.createPool({ }) 创建连接池，使用connectionLimit设置数量，默认是15个  pool.query(sql,[],callback) 执行SQL语句，sql执行的SQL语句，第2个参数用于替换SQL语句中的值，callback回调函数，获取执行的结果 |

练习：使用连接池中的连接向员工表插入数据

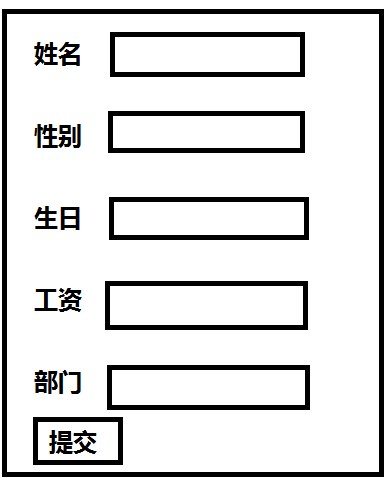
练习：创建web服务器，托管资源到public目录下，创建文件add.html,点击提交，向服务器发请求get,/add,服务器端获取该数据



课后任务：

（1）复习，整理思维导图

（2）练习：创建web服务器，托管静态资源，创建emp\_add.html,点击提交,将数据添加到tedu下的emp表中。点击提交请求 get /add



# day06

mysql连接方式

普通连接

连接池 创建固定连接，使用效率高

SQL注入 ‘select \* from emp where eid=?’,[5]

任务：完成学子商城的商品模块

列表 详情 删除 修改 添加