目录

[ node.js 1](#_Toc491178355)

[1、 计算程序运行时间：循环计时 2](#_Toc491178356)

[2、 模块-文件模块+目录模块（Module） 2](#_Toc491178357)

[3、 querystring 将查询字符串转换js对象 4](#_Toc491178358)

[4、 url模块（地址字符查询-）将请求url地址转js对象 4](#_Toc491178359)

[5、 Buffer模块 5](#_Toc491178360)

[6、 fs模块-同步和异步读取/写、追加文件 5](#_Toc491178361)

[7、 http-模块（服务器模块）：详情day03(05\_http例子) 6](#_Toc491178362)

[8、 node.js第三方模块：（MySQL）-npm工具下载 7](#_Toc491178363)

[9、 连接池（提高效率）node.js连接mysql 7](#_Toc491178364)

[10、 express模块，快速开发http服务器 8](#_Toc491178365)

[11、 静态文件中间件(函数)-处理自动完成public内的文件 9](#_Toc491178366)

## node.js

静态网页和动态网页

**静态**:网页内容任何人在任何时间访问都不变的

HTML/CSS/JS/FLASH/音频/视频

**动态:** 网页内容任何人在不同的时间访问可能不同的

JSP/PHP/ASP/NODEJS

JSP =HTML+JAVA 功能强大可靠，适合大型企业级项目

PHP =HTML+PHP 简单易用，适合互联网项目

ASP.NET =HTML+C# 易用，windows平台

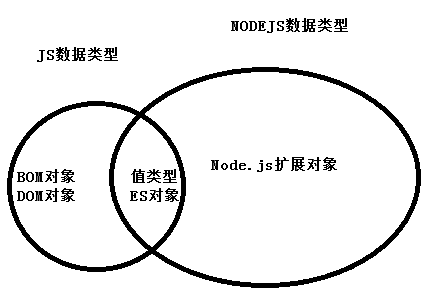
Node.JS =性能好，适合于IO密集型项目，不适合

CPU密集型项目

1亿=8小时\*3600=28800/10= 347/s

\*IO密集型项目:(input output)查询、删除、更新

\*CPU密集型项目:天气预报,滴滴打车

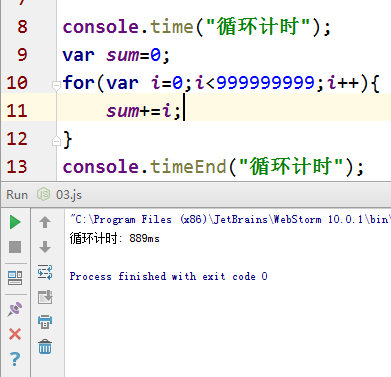


### 计算程序运行时间：循环计时

console.time(“循环计时”)； //开始计时

function; //…..运行的程序；

console.timeEnd(“循环计时”)； //结束计时，循环计时的内容要一致



### 模块-文件模块+目录模块（Module）

* Modal:模态框
* Model:模型
* Module:模块

Node.JS按照功能不同，可以把函数，对象分别放到不同文件、目录下、这些**文件/目录**NodeJS中就称为模块"Module"（如商品管理;用户管理模块;支付模块;）

Node.js中每个模块都是一个独立构造函数，解释器会为每个.js文件Node.js添加如下代码.

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

exports:{} 用于声明向外导出自己的成员

require:fn 用于导入其它的模块，并且创建指定模块对象

module: 当前模块对象

\_\_filename 文件绝对路径

\_\_dirname 文件所属目录绝对路径

//自己编写的文件内容

});

* **自定义文件模块：**

创建一个js文件,如m3.js 导出需要公共数据,其它模块可以require("./m3") 模块

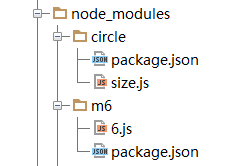
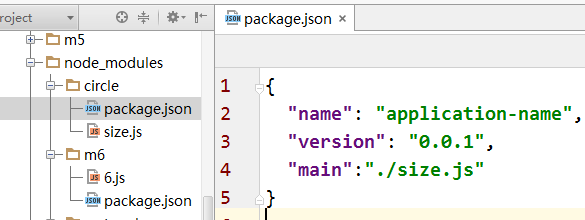
1.db.js:

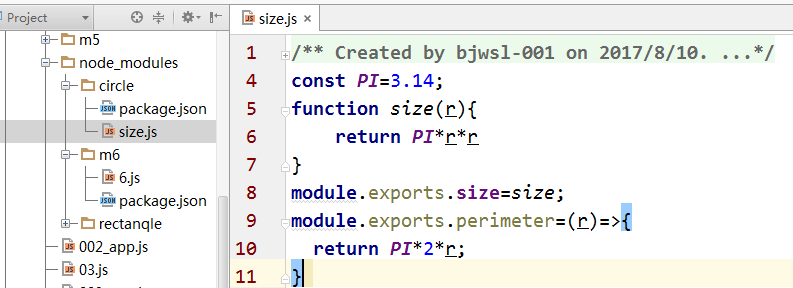
**var *uname***=**"james"**;  
**function** *run*(){  
 **console**.log(**"奔跑的"**+***uname***)  
}  
*//对象向外公开变量和函数*exports.**run**=*run*;

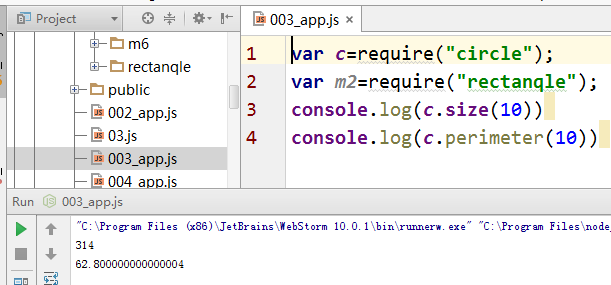
2.user

**var *m***=require(**"./db"**);  
***m***.**run**();  
*//console.log(m.a);*

* **自定义目录模块：**





### querystring 将查询字符串转换js对象

*//功能:处理查询字符串  
//qs.parse(str); 将查询字符串转换js对象  
//qs.stringify(obj);js对象转换查询字符串  
//加载其它模块:通常不需要修改只使用***const *qs*** = require(**"querystring"**);  
**var *str*** = **"uname=tom&upwd=123&age=19"**;  
*//把客户端传递给 node.js服务器查询字符串解析  
//容易使用JS对象***var *obj*** = ***qs***.parse(***str***);  
**console**.log(***obj***);  
  
*//把一个JS对象转抡为查询字符串格式***var *obj2*** = {**ename**:**"强东"**,**age**:20,**addr**:**"bj"**};  
***str*** = ***qs***.stringify(***obj2***);  
**console**.log(***str***);  
*//结果：ename=%E5%BC%BA%E4%B8%9C&age=20&addr=bj*

### url模块（地址字符查询-）将请求url地址转js对象

* 1. **const *url*** = *require*(**"url"**);  
     **var *u*** = **"** **http://127.0.0.1:8081/add.do?sname=dsaf&scope=98"**;  
     **var *obj*** = ***url***.parse(***u***); //转为一个对象  
     *//console.log(obj);  
     //true:指定对URL中的查询字符串进行进一步解析,  
     //解析查询字符串:*

Url {

protocol: null,

slashes: null,

auth: null,

host: null,

port: null,

hostname: null,

hash: null,

search: '?sname=dsaf&scope=98',

**query: { sname: 'dsaf', scope: '98' },**

pathname: '/add.do',

path: '/add.do?sname=dsaf&scope=98',

href: '/add.do?sname=dsaf&scope=98' }

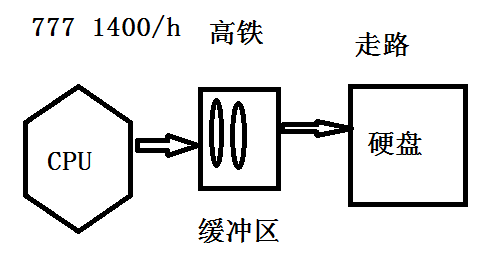
**var** path=obj.**pathname**; *//请求路径不带查询字符串*

**console.log(path) // 结果：**//add.do

**var *obj2*** = ***url***.parse(***u***,**true**);

***console***.log(***obj2***.**query**.**uname**); //结果：dsaf

### Buffer模块



分配一个指定大小的缓冲区

var buff1 = Buffer.alloc(1024)

**var *buf1***=Buffer.alloc(10);  
**console**.log(***buf1***);

使用一个数字数组创建一个缓冲区

var buff2 = Buffer.from([1,2,3]);

使用一个字符中创一个缓冲区

var buff3 =Buffer.form("abc");

把一个缓冲区中的数据转换字符串

var str = buffe3.toString();

### fs模块-同步和异步读取/写、追加文件

fs(file system)模块提供了对文件系统中文件/目录进行增删改查，读写的功能.

原生模块--fs =》const fs=require("fs");

**同步：**

* + 读取文件内容：**readFileSync(file)**

**const** fs = require(**"fs"**);  
console.log(**"开始文件写开始。。。"**);  
**var** str=**new** Date().toString();  
**var** file=**"./public/2.log"**;  
fs.writeFileSync(file,str);  
console.log(**"开始文件写结束..."**);

* + 写入文件：**writeFileSync(file,str/buff)；**
  + 向文件添加内容：**appendFileSync(file,str/buff);**

**异步：**

* + 异步读取文件中的内容

**fs.readFile(file,function(err,data){}) //file☞文件地址**

* + 异步向文件写内容

**fs.writeFile(file,str/buf,function(err){})**

*//异步（asynchronize）读取文件内容***const *fs***=require(**"fs"**);  
**console**.log(**"1、文件读取开始"**);  
**var *src***=**"./public/4.css"**;  
**var *des***=**"./public/44.backup.css"**;  
***fs***.readFile(**"./public/4.css"**,**function**(err,data){

//文件读取完成调用该函数  
 **if**(err)**throw** err;  
 **console**.log(**"文件读取完成"**);  
 ***fs***.writeFile(***des***,data,**function**(err){  
 **if**(err)**throw** err;  
 **console**.log(**"文件复制完成"**)  
 })  
});  
**console**.log(**"3:js执行完成"**);

### http-模块（服务器模块）：详情day03(05\_http例子)

网站正在升级：（指定输出编码utf-8）

**const *http***=require(**"http"**);  
**const *fs***=require(**"fs"**);  
**var *server***=***http***.createServer();  
***server***.listen(8081);  
***server***.on(**"request"**,(req,res)=>{  
 res.setHeader(**"Content-type"**,**"text/html;charset=utf-8"**);  
 res.write(**"<h1>您好！网站正在升级中请5秒后重试</h1>"**)  
 res.end(); //通知客户端响应消息结束；  
})

### node.js第三方模块：（MySQL）-npm工具下载

**const *mysql***=*require*(**"mysql"**);  
**var *conn***=***mysql***.createConnection({  
 **host**:**"127.0.0.1"**,  
 **user**:**"root"**,  
 **password**:**""**,  
 **database**:**"jd"**,  
 **port**:3306  
});

* ***//使用占位符解决安全问题 //防var u = "' drop database mysql#";*var *sql***=**"INSERT INTO jd\_user VALUES(null,?,?)"**;  
  ***conn***.query(***sql***,[**"tom3"**,**"123"**],(err,result)=>{  
   **if**(err) **throw** err;  
   **console**.log(result);  
   **if**(result.**affectedRows**>0){  
   **console**.log(**"添加成功"**)  
   }**else**{  
   **console**.log(**"添加失败"**)  
   }  
   ***conn***.end(); //关闭连接  
  })

### 连接池（提高效率）node.js连接mysql

**const *mysql***=*require*(**"mysql"**);  
**var *pool***=***mysql***.createPool({  
 **host**:**"127.0.0.1"**,  
 **user**:**"root"**,  
 **password**:**""**,  
 **database**:**"jd"**,  
 **port**:3306,

* **//连接池**  
   **connectionLimit**:10 *//连接池大小；*})  
  ***pool***.getConnection((err,conn)=>{  
   **if**(err)**throw** err;  
   **var** sql=**"UPDATE jd\_user SET upwd=? WHERE uid=?"**;  
   conn.query(sql,[**"111"**,2],(err,result)=>{  
   **if**(err)**throw** err;  
   **if**(result.**affectedRows**>0){  
   **console**.log(**"更新成功"**)  
   }**else**{  
   **console**.log(**"更新失败"**)  
   }  
   });  
   conn.release(); *//归还连接*  
  });

### express模块，快速开发http服务器

该模块还有自己专用官网: http://expressjs.com

该模块中文镜像网站:http://www.expressjs.com.cn

也可以npmjs.org 上下载模块

npm install express

npm i express

*//1:加载相应的模块***const *http*** = *require*(**"http"**);  
**const *express*** = *require*(**"express"**);  
**const *mysql*** = *require*(**"mysql"**); *//insert***const *qs*** = *require*(**"querystring"**);*//post 参数  
//2:创建express对象***var *app*** = ***express***();  
*//3:创建服务器对象***var *server*** = ***http***.createServer(***app***);  
*//4:绑定监听端口****server***.listen(8081);  
*//5:创建连接池***var *pool*** = ***mysql***.createPool({  
 **host**:**"127.0.0.1"**,  
 **user**:**"root"**,  
 **password**:**""**,  
 **database**:**"dangdang"**,  
 **port**:3306,  
 **connectionLimit**:10  
});  
*//6:处理二个应用  
//get /book\_add.html****app***.get(**"/book\_add.html"**,(req,res)=>{  
 res.sendFile(***\_\_dirname***+**"/public/book\_add.html"**);  
});  
*//post /bookadd.do****app***.post(**"/bookadd.do"**,(req,res)=>{  
 req.on(**"data"**,(data)=>{  
 **var** str = data.toString();  
 **var** obj = ***qs***.parse(str);  
 ***console***.log(obj.**title**+obj.**price**);  
  
 ***pool***.getConnection((err,conn)=>{  
 **if**(err)**throw** err;  
 **var** sql = **"INSERT INTO book VALUES(null,?,?,now())"**;  
 conn.query(sql,[obj.**title**,obj.**price**],(err,result)=>{  
 **if**(err)**throw** err;  
 **if**(result.**affectedRows**>0){  
 res.json({**code**:1,**msg**:**"添加成功"**});  
 }**else**{  
 res.json({**code**:-1,**msg**:**"添加失败"**});  
 }  
 conn.***release***();  
 })  
 });  
 *//res.send("ok");* });  
});

### 静态文件中间件(函数)-处理自动完成public内的文件

//6:处理所有静态文件

//express提供非常实用功能:静态文件中间件(函数)

//你指需要指定(目录) public

//该目录下所有静态资源请求，读取，发送，帮助我们自动完成

//示例:首页 http://127.0.0.1:8081/index.html

**app.use(express.static("public"));**

### 结果返回主键增长的id:result.insertId;

//第一个Sql发送成功，结果返回id给第二个sql用：

***pool***.getConnection((err,conn)=>{  
 **if**(err)**throw** err;  
 *//3:创建SQL insert并且发送SQL* **var** sql1 = **"INSERT INTO jd\_user VALUES(null,?,?)"**;  
 conn.query(sql1,[uname,upwd],(err,result)=>{  
 **if**(err)**throw** err;  
 **var** uid = result.**insertId**;  
 **var** sql2 = **"INSERT INTO jd\_userinfo VALUES("**;  
 sql2+= **"null,?,?,?,?,?)"**;  
 conn.query(sql2,[hp,email,age,rd,uid],(err,result)=>{  
 res.json({**code**:1,**"msg"**:**"添加成功"**});  
 conn.release();  
 });  
 });