nodejs/

1.nodejs概述

nodejs基于谷歌v8引擎，运行在服务器端端的语言，基于JS

对比JS和nodejs

（1）js运行在及客户端浏览器，存在多个浏览器，容易产生兼容性的问题；nodejs在服务器端只有一个环境，不存在兼容性

（2）两者都有（ES）对象，自定义对象、宿主对象（根据执行环境的不同划分）

（3）JS用于网页中的交互效果，Nodejs用于服务器操作。例如web服务器创建，数据库存在，文件操作等

http://www.nodejs.org英文官网

http://www.nodejs.cn中文网

（4）nodejs的特点

单线程逻辑处理（占一个cpu,每次处理一件事）

非阻塞的异步I/O（输入/输出）处理

支持数万个并发连接

nodejs的应用场景

基于社交网络的大规模web应用

nodejs不适合CPU的密集型的应用

递归、数据加密解密、数据挖掘和数据分析

（5）nodejs的执行方式

脚本模式 node c/xampp/…/1.js

交互模式 node 回车 进入交互模式

退出 两次ctrl+c或者 .exit 退出交互模式

2、全局对象

全局作用域下的变量就是全局对象下的属性，全局对象下的函数就是全局对象下的方法，可以使用全局对象来访问。

nodejs:global.

> var a=1;

undefined

> global.a

1

在交互及模式下，声明的变量和函数都属于全局下的，可以使用global来访问。

在脚本模式下，声明的变量和创建的函数都不属于全局下的，一个文件默认会创建一个独立的作用域，叫做文件（模块）作用域， 防止污染全局。

js.window

在浏览器下，文件中声明的变量或者创建的函数都属于是全局作用域下的，会污染全局。

（1）console对象---控制台

global.console.log(); //打印日志

global.console.info();//打印消息

global.console.warn();//打印自定义的警告

global.console.error();//打印自定义的错误

global.console.time();//开始计时

global.console.timeEnd();//结束计时

（2）process对象——进程

process.arch//当前计算机CPU

process.platform//查看当前操作系统

process.env//查看当前系统所有环境变量

process.version//查看当前nodejs版本号

process.pid//查看当前进程编号

process.kill()//结束某个编号的进程

（3）Buffer对象——缓冲区

一块用于临时存储数据的内存区域，可以存储文件数据，网络上传输的资源（视频、在线直播…）

创建Buffer

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

buffer数据<Buffer 61 62 63 64 65>

将buffer数据转为字符串

buf.toString()

自定义字符串前后保持一致

//检测程序的耗时

global.console.time('for');//开始计时

for (var i=1;i<=100000 ;i++ ){

}

global.console.timeEnd('for');//结束计时

练习：计时查看，for、while、dowhile循环10000的耗时

global.console.time('for');//开始计时

for (var i=1;i<=10000 ;i++ ){

}

global.console.timeEnd('for');//结束计时

var i=1

global.console.time('while');

while (i<=10000)

{i++

}

global.console.timeEnd('while')

var i=1

global.console.time('dowhile')

do

{i++}

while (i<=10000);

global.console.timeEnd('dowhile')

3、模块

模块就是一个封装好的功能体

在nodejs下模块分为自定义模块、核心模块（nodejs官方提供）、第三方模块（npm网页中可查）

（1）自定义模块

在nodejs下，任意一个js文件都是一个模块，文件中的代码默认被一个构造函数所包含

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

//程序员编写的代码{

})

exports//等价于module.exports

require（）//引入一个模块

//引入模块419-6.js

// ./ 同一级目录

require('./419-6.js')

module//当前的模块对象（module.exports导出的对象，包含供其他模块使用的属性和方法）

console.log('419-6.js')

var a=1;

//导出：公开那些变量或者函数

//默认导出的是一个空对象

//添加导出的内容，往导出的对象中添加属性和方法

//module代表当前的模块对象

//module.exports导出的对象

module.exports.st=a;

//引入模块419-6.js

// ./ 同一级目录

//把419-js模块导出的结果放入到变量中

var obj=require('./419-6.js')

console.log(obj);

练习：在419-6.js声明变量b,创建函数fn，导出变量和函数

//引入模块419-6.js

// ./ 同一级目录

//把419-js模块导出的结果放入到变量中

var obj=require('./419-6.js')

obj.fn;

//console.log(obj);

\_\_filename//当前模块的完整路径和文件名称

\_\_dirname//（dir:directory目录）当前模块的完整路径

作业：创建两个模块main.js(主模块)，circle.js(功能模块)，在功能模块中创建两个函数，传递一个参数半径，计算圆的周长和面积，到处这两个函数；在主模块中引入功能模块，并调用两个方法

function str1(r){

return 2\*Math.PI\*r;

}

function str2(r){

return Math.PI\*Math.pow(r,2)

}

/\*module.exports.str1=str1

module.exports.str2=str2

var circle=require('./4-22.js')//模块的后缀名.js可省略

console.log(circle)

console.log(circle.str1(5).toFixed(2))

console.log(circle.str2(5).toFixed(2))

模块分类

|  |  |  |
| --- | --- | --- |
|  | 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  果后缀名是js，则可可以省略 | require('url')  用于引入官方提供的核心模块 |
| 目录模块 | require('./02-2')  在02-2目录下寻找package.json中寻找main属性对应的文件，找不到会到会去引用index.js; | require('04-2')  自动到当前目录下node\_modules中寻找目录模块04-2，如果找不到会到上一级目录下寻找，直到顶级目录。常用于第三方模块 |

练习：创建模块03-1.js，引入当前目录下的03-2目录模块；在03-2下创建test.js，导出一个函数fn（计算两个数字相加），在03-1.js调用

练习：在05目录下创建模块05-1.js,引入不带路径模块05-2在05-2下含有hello.js文件（打印一句话）

4、npm和包

包（package）:含有一个目录模块，里面包含多个文件，其中有一个文件名为package.json,是包说明文件，包含有包的信息

npm:包管理工具，安装nodejs的时候会附带安装，用于管理包，包括下载，上传升级…

npm官网:www.npmjs.com

在cmd命令行输入：npm -v查看

npm install mysql回车

①切换目录

cd 完整路径

如果要进入其他的盘符

d:

进入指定的目录下，在空白区域按住shift键，单击右键🡪在此处打开powershell

②使用npm下载安装第三方包

npm install 包名称

③生成package.json

npm init -y 自动生成一个package.json文件，后期使用npm安装的包都会记录到这个文件中

npm install 自动安装package.json 文件中记录的包

5、全局函数

parseInt/parseFloat/ecodeURI/decodeURI/isNaN.

isFinite/eval

(1)一次性定时器

开启

var timer=setTimeout（回调函数，间隔时间）；

当间隔的时间到了，执行回调函数，单位是毫秒。

清屏：clearTimeout（timer）

（2）周期性定时器

开启

var timer=setInterval(回调函数，间隔时间)

每隔一段时间，执行一次回调函数，

清楚

clearInterval(timer)

//开启一次性定时器

var timer=global.setTimeout(function(){

console.log('炸弹爆炸')

},3000)

//清楚一次性定时器

clearTimeout(timer)

//开启周期性定时器

var timer=global.setInterval(function(){

console.log('炸弹爆炸')

},1000)

//清楚一次性定时器

clearInterval()

练习：使用周期是哪个定时器每隔三秒钟打印hello,打印三次以后，清除定时器。

变量是打印次数，。如故次数为三，清除定时器

var i=0;

var ter=global.setInterval(function(){

console.log('哈哈',3000)

i++;

if(i==3){clearInterval(ter)}},3000)

（3）立即执行定时器

var time=setImmediate(回调函数)

回调函数会放入到队列中

当主线程程序执行完，才会执行队列中的内容

清除

clearImmediate(time)

var str=setImmediate(function(){

//回调函数会放入到队列中

//当主线程执行完，才会执行队列的内容

//console.log('你好吗')

console.log('查询数据库')

})

clearImmediate(str)

process.nextTick(回调函数)（服务器端运行）

在主程序的末尾执行

//在主程序执行完立即执行

process.nextTick(function(){

console.log('操作数据')

})

console.log('连接数据库')

6、querystring模块-----查询字符串

属于核心模块，nodejs官方提供的。可以直接引入，不需要创建

querystring

www.codeboy.com/detal.html?lid=5&name=dell

浏览器向服务器发送请求，传递数据的一种方式

parse()将查询字符串格式化为对象

stringify()将对象转换成查询字符串

/\*//引入querystring模块

const querystring=require('querystring');

//查询字符串

var str='lid=5&name=dell'

//使用查询字符串模块，将查询字符串格式化对象

var obj=querystring.parse(str)

console.log(obj)

//将对象转换成查询字符串

var str2=querystring.stringify(obj);

console.log(str2)\*/

练习：获取百度查询中查询字符串中的关键字

e=utf-8&tn=baidu&wd=电脑

const querystring=require('querystring');

var str='e=utf-8&tn=baidu&wd=电脑'

var obj=querystring.parse(str)

console.log(obj)

var str2=querystring.stringify(obj);

console.log(str2)

7、url模块

parse()将url格式化为对象

protocol协议

hostname服务器域名/ip地址

port端口号

pathname文件在服务器上的路径

query查询字符串

format()将对象转成url

query属性对应的是对象

//引入url模块

const url=require('url');

//url

//var str='http://www.codeboy.com:80/product/detail.html?lib=5&name=dell';

//将url格式化为对象

//var obj=url.parse(str)

//console.log(obj.query)

const url=require('url');

//将对象转换成url

var obj={

protocol:'http',

hostname:'176.163.0.224',

port:8080,

pathname:'/admin/login.html',

query:{uname:'root',

upwd:'123456'

}

}

var str=url.format(obj)

console.log(str)

练习：获取浏览器中传递数据

http://www.tmooc.cn:3000/web/1903.html?sid=10&course=javascript

获取url中查询字符串

将查询字符串格式化为对象

//引入url模块

const url=require('url')

//引入查询字符串模块

const querystring=require('querystring')

var str='http://www.tmooc.cn:3000/web/1903.html?sid=10&course=javascript'

//获取url中查询字符串

var obj=url.parse(str)

var qs=obj.query

//将查询字符串格式化为对象

var obj2=querystring.parse(qs)

console.log(obj2.sid,obj2.course)

8、fs模块---文件系统模块

用于文件的操作，目录的创建，删除，读取；文件的创建，读取写入，删除

（1）查看文件的状态

fs.stat(path,callback)

path 要查看的文件路径

callback 回调函数，用于获取文件的状态

err如果查看失败错误的信息

stats 文件的状态信息

isDirectory 是否为目录

isFile() 是否为文件

//引入文件系统模块

const fs=require('fs')

//查看文件的状态

//参数1：要查看的文件路径

//参数2：回调函数，用于获取结果

fs.stat('08-url.js',function(err,stats){

//err:如果查看失败的错误信息

//throw抛出错误,组织往后执行

if (err) throw err;

//stats:文件的状态信息

console.log(stats)

//是否为目录

console.log(stats.isDirectory())

//是否为文件

console.log(stats.isFile)

});

练习：复习今天内容

（2）创建模块01-1.js，引入02-2不带路径的目录，在02-2中创建文件file.js,在file.js中，导出一个查询字符串，在01-1.js中获取该字符串，并获取其中的数据

同步和异步

同步：statSync(path)方法调用后，必须等待方法结束才能继续后边的任务。通过返回值获取结果。（耗时）

异步：fs.stat(path,callback)方法调用后无须等待结束，直接去执行后续的代码，整个过程不会阻碍后边的任务；通过回调函数来获取结果。（自上而下执行，快速执行；）

//引入模块

//const fs=require('fs')

/\*//异步方法

//fs.stat('01\_homework',function(err,stats){

//if (err)throw err;

//console.log(stats)

//})

//同步方法

//把查看的文件状态保存在到变量中

//使用方法的返回值获取结果

//新建一个文件01\_homework

var stats=fs.statSync('01\_homework')

console.log(stats);\*/

9、fs模块---文件系统

（1）查看文件状态

const fs=require('fs')

fs.stat(path,callback)/statSync(path)01

（2）创建目录

const fs=require('fs')

fs.mkdir(path,callback)/mkdirSync(path)

fs.mkdirSync('mkdir3')

（3）移除目录

const fs=require('fs')

fs.rmdir(path,callback)/rmdirSync(path)

fs.rmdir('mkdir3',function(err){

if(err) throw err;

console.log('移除成功')

})\*/

或者

fs.rmdirSync('mkdir3')

console.log(1)

（4）读取 目录中的内容

fs.readdir(path,callback)/reddirSync(path)

callback

err 可能产生的错误

files 读取的结果，格式为数组

const fs=require('fs')

fs.readdir('01\_homework',function(err,files){

if(err) throw err;

//files是读取的结果

console.log(files)

})

const fs=require('fs')

files=fs.readdirSync('01\_homework')

console.log(files)

console.log(1)

(5)清空写入文件

写入文件

fs.writeFile(path,data,callback)/writeFileSync()

path 要写入的文件路径

data 要写入的数据

callback 回调函数

如多文件不存在会创建文件，如果文件已经存在，会清空文件中的内容，然后写入

（6）追加写入

fs.appendFile(path,data,callback)/appendFileSync()

如果文件存在，在文件的末尾写入数据

练习：创建数组，包含有多个姓名， 便利数字获取每个元素，把每个元素使用同步方法写入到文件data.txt

const fs=require('fs')

var ename=['hhh','lili','shaha','make']

for (var i=0;i<ename.length;i++)

{fs.appendFileSync('data.txt',ename[i]+'\n')

}

console.log(1)

（7）读取文件

fs.readFile(path,callback)/readFileSync()

callback

err 可能产生的错误

data 读取的文件中的数据，格式为buffer

const fs=require('fs')

fs.readFile('1.txt',function(err,data){

if(err) throw err;

console.log(data.toString())

})

console.log(1)

(8)删除文件

fs.unlink(path,callback)/unlinkSync()

const fs=require('fs')

fs.unlink('1.txt',function(err){

if(err) throw err;

console.log('删除成功')

})

console.log(1)

(9)判断文件是否存在

fs.existsSync(path)

如果存在会返回true,不存在返回false

const fs=require('fs')

var str=fs.existsSync('data.txt')

console.log(str)

console.log(1)

练习：判断文件2.TXT是否存在，如果存在删除

判断目录mydir2是否存在，如果不存在创建

const fs=require('fs')

if( fs.existsSync('2.txt')){

fs.unlink('2.txt',function(err){

if(err) throw err;

console.log('删除成功')})

}

const fs=require('fs')

if (!fs.existsSync('mydir2'))

{fs.mkdirSync('mydir2')

console.log('创建成功')

}

console.log(1)

10、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，对应浏览器地址栏的内容，要向服务器获取哪些内容

Request Method:请求的方法，GET/POST…,获取内容的方式

Status Code:相应的状态码，可以查看到相应的结果

1\*\*：正在请求，没有结束

2\*\*：成功的响应

3\*\*：响应的重定向，跳转到另一个url；通常结合响应头信息中Location一起使用

4\*\*：客户端错误

5\*\*：服务器错误

Remote Address:请求的服务器的IP地址和端口号

(2)响应头信息（ResponseHeaders）

Cconnection:连接的方式，keep-alive持续连接

Content-Type:相应的文件类型

Content-Length:响应的文件长度

Transfer-Encoding:传输方式，chuncked分段传输

Content-Encoding:压缩方式gzip

Location:当响应重定向的时候，跳转的url

(3)请求头信息（RequestHeaders）

Accept:客户端接受的文件类型有哪些

Accept-Encoding:客户端接受的压缩形式

User-Agent:客户端发请求使用的浏览器

(4)请求主体

可有可无，客户端向服务器端传递数据

11、http模块

既可以模拟浏览器向web服务器发请求，也可以创建web服务器

（1）模拟浏览器

http.get(url,callback)

get请求的办法

callback 回调函数，用于获取服务器端响应

res响应的对象

statusCode 响应的状态码

res,on('data',function(buf){ })

通过事件获取服务器端的响应，当有数据传输的时候自动触发；通过回调函数来接受数据；buf就是传输的数据格式buffer

//引入http模块

const http=require('http')

//模拟浏览器向web服务器发请求

//get:请求的方法

//参数1：请求的URL

//参数2：回调函数，用于获取服务端响应

http.get('http;//www.codeboy.com',function(res){

//res：相应的内容，格式为对象

//console.log(res)

//响应的状态码

console.log(res.statusCode)

//响应的数据

//通过一个事件获取：当有数据传输的时候，自动触发这个事件

//也是使用回调函数来获取

res.on('data',function(buf){

console.log(buf.toString());

})

});

练习：使用http模块下的get方法向http://www.tmooc.cn发送请求，并获取相应的数据

const http=rquire('http')

http.get('http://www.tmooc.cn',function(res){

console.log(res.statusCode)

res.on('data',function(buf){

console.log(buf.toString())

})

})

(2)创建web服务器

var server=http.createServer()创建web服务器

server.listen(8080)设置web服务器端口号，监听端口号变化

server.on('request',function(req,res){ })

接收浏览器的请求，是一个事件，当有请求发生自动触发

req 请求的url，显示端口号

method请求的方法

res响应的对象

writeHead(code,obj)设置响应的状态码和头信息

code状态 obj是一个头信息

write() 设置响应的内容

end ()结束响应并发送到浏览器

练习使用http模块创建web服务器，设置端口8080；接收浏览器的请求

/login 响应 this is login page

/member 响应 welcome to home

/ 重定向 /login

以上都没有响应内容404 not found

const http=require('http')

var server=http.createServer()

server.listen(8080)

server.on('request',function(req,res){

if(req.url=='/login'){

res.write('this is login page')}else if(req.url=='/member'){

res.write('welcome to home')

}else if(req.url=='/'){res.writeHead(302,

{Location:'http://www.tmooc.cn'})}else{

res.write('404 not found')}

res.end()

})