NodeJS

Nodejs是基于谷歌V8引擎解释器,运行在服务器端的语言.本质上还是JS

对比JS和Nodejs

JS运行在浏览器端,因为有多款浏览器,每个浏览器都有自己的解释器,所以存在代码兼容性的问题

NODEJS 在服务器端只有一个运行环境,不存在兼容性问题

二者都有内置(ES)对象,自定义对象,宿主对象(根据执行环境的不同)

JS用于网页上的交互效果,NODEJS用于服务器端数据库操作,文件操作

Nodejs两种执行模式

1,脚本模式 node 路径

2,交互模式

Node 回车 进入交互模式

两次control +C 或者.exit

3,Nodejs应用场景

支持数万个并发连接,适用于一些基于社交网络的大规模web应用

4,全局函数

Global

在一个文件中声明的变量或者常见函数,都不是全局对象下的,一个文件也可以称为文件作用域(局部作用域)

在交互模式下.声明的变量或者函数都是全局对象

Js:window

在浏览器下,文件下声明的变量或者创建的函数都属于全局对象下的变量或者函数

Console对象:

global.console.log(123); 打印消息

global.console.info(123);打印消息

global.console.error(123);打印错误消息 ❌

global.console.warn(123);打印警告消息 ⚠️

//计算for循环的耗时

global.console.time('for循环');//计时开始

for (let i = 0; i <= 100000; i++){

}

global.console.timeEnd('for循环');//计时结束

//while循环计时

var i = 0;

global.console.time('while循环');

while (i<=100000) {

i++;

}

global.console.timeEnd('while循环')

//do while循环计时

var a = 0;

global.console.time('dowhile循环');

do {

a++;

} while (a <= 100000)

global.console.timeEnd('dowhile循环');

2,process对象

Process.arch 查看当前的CPU架构

Process.platform 查看当前的操作系统

Process.env 查看当前计算机的环境变量

Process.version 查看nodejs的版本号

3,buffer 对象

缓冲区,在内存中存储数据的区域.存储网络传输时的资源

创建buffer

Buffer.alloc(5,’abcde’);

将Buffer中存储的数据转为普通的字符串

Buffer.toString();

4,全局函数

可以使用global来调用

perseInt/parseFloat/encodeURI/decodeURI/eval/isNAN/isFinite..

课后任务

1,复习今天的内容,整理思维导图

2,课后练习:

双色球抽奖原理

在1-33随机取6个数,不重复;在1-16随机取1个数,第二次取得值要放在最后

Day02

全局函数

1,一次性定时器

开启

Var timer setTimeout(()=>{},) (回调函数,间隔时间)

清除

clearTimeout(timer);

//1,一次性计时器

//开启,把开启的定时器放入到变量中,便于清除

var timer = global.setTimeout(() => {

console.log('boom!');

}, 3000);

//清除一次性计时器

clearTimeout(timer);

2,周期性定时器

Var timer = setInerval(()=>{},)(回调函数,周期间隔时间)

claerInterval(timer)

var timer = setInterval(() => {

console.log('闹钟响了,快起床');

}, 3000)

//清除周期性计时器

clearInterval(timer);

//练习使用周期性计时器,每隔3秒打印hello,打印3次后清除

//声明变量,统计次数

var i = 0;

var timer = setInterval(() => {

console.log('hello');

//打印一次,i+1

i += 1;

//打印3次,清除计时器

if (i == 3) {

clearInterval(timer);

}

}, 3000)

3,立即执行

process.nextTick(()=>{})

一个事件循环的最后

4,立即执行

setImmediate(()=>{})

一个新事件循环的开始

清除

clearImmediate(定时器名称)

模块

Nodejs下有大量的模块,模块可以理解为一个功能体

分为自定义模块,核心模块,第三方模块(组织或者个人发布的)

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

*//任意一个模块中,我们写的代码都默认被一个构造函数所包含*

*//console.log(\_\_filename);//文件的完整路径和文件名称*

*//console.log(\_\_dirname);//文件的完整路径(没有文件名称)*

*//当前模块中的变量,函数都属于是局部作用域下的,不能被外部使用,如果要被外部访问,就需要将它导出*

var a = 1;

var b = 'hello';

module.exports.num = a;

module.exports.str = b;

*//console.log(module.exports); //要导出的内容--对象*

*//})*

module.exports 当前模块导出的对象,可供其他的模块使用当前模块中的属性和方法

exports 和module.exports 的作用一样

require 用于引入一个模块

/\*这是一个矩形模块,其中有两个变量,分别是宽度width/height

有两个方法,获取面积getArea(),

获取周长getLength()

创建07\_main.js,引入上一个模块,分别输出圆的周长和面积

\*/

//被引用模块

var width = 20, height = 10;

function getArea() {

//console.log(`面积为${width\*height}`)

return width \* height;

}

function getLength() {

//console.log(`周长为:${(width+height)\*2}`)

return 2 \* (width + height);

}

module.exports.getArea = getArea;

module.exports.getLength = getLength;

//引用模块

var res =require('./07\_rect.js');

console.log('面积为 '+res.getArea()) ;

console.log('周长为 '+res.getLength()) ;

文件模块 require(‘./07\_rect.js’)用于用户自定义的模块.后缀名是js 可以省略

不以路径开头 require(‘querystring’)通常用于引入nodejs官方提供的核心模块

目录模块 require(‘./08\_1’) 在目录下要存在一个index.js文件

或者使用package.json文件声明属性main,来指定引入的文件

注意:如果引入目录和文件名称相同,则优先引入目录,如果目录不存在,引入文件

不带路径的目录 require(‘10\_3’) 要求查找的目录放到当前目录下的node\_modules中,如果不存在,则到上一级的mode\_modules中查找,直到查找到最顶层,通常用于查找第三方模块

3,包和npm

包(package):就是一个目录模块,里边含有多个文件,其中包含有一个命名为package.json的文件.是包的说明文件

NPM: Node Package Manage, nodejs的包管理工具

使用npm可以下载第三方的包,默认会到[www.npmjs.com](http://www.npmjs.com)去下载所要安装的包 npm install 包名称

Day03

1 核心模块

是Nodejs官方提供的模块,可以直接引入,不需要创建

1.1查询字符串

浏览器像服务器发请求的时候,所传递的数据,也称为参数

Parse() 用于将查询的字符串解析成对象

Stringify() 用于将对象转成查询字符串

1.2 url模块

Parse() 将网址解析为对象

解析的结果中

Protocol 协议

hostname 主机

port 端口

pathname 文件在服务器上的路径

query 查询字符串

hash 哈希(锚点链接)

url.format() 将对象转换为网址格式

query 查询字符串 得为对象的模式 query:{}

//练习:浏览器端请求的网址http://www.tedu.cn/web/1809.html?sid=20&name=tom#detail

//获取查询字符串中的sid和name两个对应的值

var str = 'http://www.tedu.cn/web/1809.html?sid=20&name=tom#detail';

const url = require('url');

const querystring = require('querystring');

var obj = url.parse(str);

console.log(obj);

//获取网址中的查询字符串

var qs = obj.query;

//把查询字符串解析为对象

var obj2 = querystring.parse(qs);

console.log(`查询结果

编号为${obj2.sid}

名字为${obj2.name}`)

2,文件系统模块—fs

2.1

Fs.stat(path,callback)查看文件的状态

Path 要查看的文件路径

Callback 回调函数.在里边有两个参数

Err 如果查看失败的的错误信息

Stats 没有错误,文件的状态信息

fs.stat('./05\_url.js', (err,stats) => {

//err 如果查询有错误,把错误信息放到这参数(err)中

if (err) {

//抛出错误,阻止后续代码执行

throw err;

}

//console.log(stats);

//是否为文件

console.log(stats.isFile());

//是否为目录

console.log(stats.isDirectory());

});

2.2 查看文件的状态 statSync

//同步 阻塞式

var res = fs.statSync('./05\_url.js');

console.log(res.isFile());

console.log(res.isDirectory());

对比同步方法和异步方法的区别

同步方法会阻止后续代码的执行,只有这个方法执行完成过后,才能继续执行后续代码;是通过返回值(放入到变量)来获取结果.

异步方法不会阻止后续代码的执行,把执行结果放到整个程序的最后;是通过回调函数来获取结果

Mkdir(path,callback){}/mkdirSync()

Rmdir(path,callback){}/rmdirSync()

Readdir(path,callback){}/readdirSync()

Path 目录的路径

Callback 回调函数

Err 读取失败的错误信息

Files 读取的文件,格式是数组

readFile(file,data,callback)/readFile(file) 读取文件

writeFile(file,data,callback)/writeFileSync(file,data) 写入文件

file 要写入的文件,data要写入的文件

如果文件不存在,则会创建文件;如果文件已经存在,则会清空文件中的内容,然后写入

appenFile(file,data,callback)/appenFileSync(file,data) 写入文件,file要写入的文件,data要写入的数据

如果文件不存在,则会创建文件;如果文件已经存在,则会在文件的末位追加写入

Unlink(path,callback)/unlinkSync(path) 删除文件

existsSync(path) 判断文件是否存在,返回true or false.

HTTP协议

1,通用头信息

Request URL:请求的url,要向服务器端请求哪个文件.对应浏览器地址栏的网址

2,相应的头信息

Request Method: 请求的方法 get/post

get 表示向服务器获取数据内容(浏览器网页)

post 表示向服务器传递一些数据(常用于注册,登录)

Status Code 响应的状态码

|  |  |
| --- | --- |
| 1\*\* | 信息，服务器收到请求，需要请求者继续执行操作 |
| 2\*\* | 成功，操作被成功接收并处理 |
| 3\*\* | 重定向，需要进一步的操作以完成请求 |
| 4\*\* | 客户端错误，请求包含语法错误或无法完成请求 |
| 5\*\* | 服务器错误，服务器在处理请求的过程中发生了错误 |

Remote Address: 远程地址

3,请求的头信息

Accept:客户端接受的文件类型有哪些

Accept Encoding:客户端接受的压缩类型

Accept Language:客户端接受的语言有哪些

Connection:连接类型 keep-alive(保持连接)

4,请求的主体

课后任务:

1,复习今天内容(核心模块),整理思维导图

2,课后练习

创建文件计时器,每次运行显示浏览的次数/

2.1 判断文件num.txt是否存在,如果不存在则创建文件,并写入初始值为0

2.2 从num.txt中读取文件的内容,将读取的数字加1,打印出来.把加1的结果再次写入(清空写入)到num.txt

全部使用同步方法