node基础

**入门**

1.什么是nodeJs？

    ·    它不是一门语言 是一门技术

    ·    也不是JavaScript框架||库

    ·    基于谷歌的v8引擎和libuv库封装的一个供js运行在浏览器之外的运行平台

    ·    node.js跳过了http服务器 速度更快 性能更牛×

    ·    node.js  单线程多进程 异步非阻塞 I/O模型

2.node.js的特点

        优点：异步I/O与事件驱动

        缺点：大量的匿名函数 加大开发和调试的难度

3.应用范围

    ·    高并发的业务     ---------》请求特别多  如直播类  电商类

    ·    I/O密集型业务   ---------》大量的读写

    ·    桌面应用

4.REPL模式

    REPL模式——Read-eval-print loop ，即输出——求值——输出循环。

打开命令提示工具，输入NODE即可进入

>

有个大于号的标识后说明进入REPL模式

命令   ctrl+c（2）：退出repl模式

          ctrl+d   退出repl模式

           tab键：查看当前的命令

           .break：退出多行表达式

           .clear:  退出多行表达式

            .save fileName  保存当前的repl模式的会话内容到指定文件

            .load fileName   载入当前repl模式 会话内容

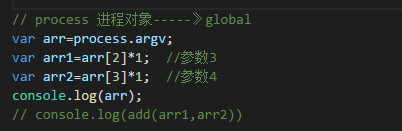
            .help 列出当前可用的命令

            \_:上一个结果

5.process 进程对象-----》global

    var arr=process.argv;

    一般前两个参数不可用 从第三个参数开始用（下标2）



6.模块

    编写稍大一点的程序时一般都会将代码模块化

    在nodejs中，每一个文件就是一个模块，而文件路径就是我们的模块名字

怎么使用模块？

在编写模块的时候，都有三个（require / exports / module）预先定义的变量可供使用

**require**

require函数 用于在当前模块中加载和使用别的模块 传入一个模块名 ，返回一个模块导出的对象

注意：模块名中的.js扩展名可以省略  内置模块没有后缀名

    相对路径

        require(./abc.js)

        require(./abc)

    绝对路径

        require(/node/abc.js)

        require(/node/abc)

**exports**

exports 对象 是当前模块的导出对象 用于导出模块的共有的方法和属性

其他模块就可以通过 require() 函数 使用当前模块时得到的就是当前模块的exports对象

model.express能抛出多个接口

NPM 包管理工具

模块 每一个js文件----》js模块

包：package -----》由多个子模块组成的大模块 称为包，并把多个子模块放在同一个目录下

npm 下载和使用 包

    npm

    初始化 npm init

    下载 npm install 包名

    下载jQuery包

     npm    init     jquery   初始化

     npm    install     jquery  -g全局     “加@版本号为下载相应版本”

**node fs模块  文件操作**

1、异步读取

　　fs.readFile( url , code , callback);

2、同步读取

　　fs.readFileSync( url , code );

IMG_257

1 var fs = require('fs');

2 // 异步读取 3 fs.readFile('input.txt', function (err, data) {

4 if (err) return console.log(err);

5 console.log('异步读取：' + data.toString());

6 })

7 // 同步读取 8 var data = fs.readFileSync('input.txt');

9 console.log('同步读取：' + data.toString());

10 console.log('程序执行完毕。');

IMG_258

3、改变文件名

　　fs.rename(old文件名，新文件名，callback(传递一个err参数))

4、同步改变文件名

　　fs.renameSync(oldPath, newPath)

1 fs.rename('hello.txt','index.txt',function(){

2 console.log("修改成功");

3 });

5、获取文件信息

　　fs.stat(path, callback) 参数使用说明如下：

　　　　path - 文件路径。 callback - 回调函数，带有两个参数如：(err, stats), stats 是 fs.Stats 对象

IMG_259

1

2 fs.stat('index.txt',function(err,stats){

3 　　if(err){

4 　　　　return err;

5 　　}

6 　　console.log(stats);

7 　　console.log("读取文件信息成功");

8 　　//检测文件类型 9 　　console.log('是否为文件(isFile) ? ' + stats.isFile());

10 　　console.log('是否为目录(isDirectory) ? ' + stats.isDirectory());

11 });

IMG_260

6、打开文件

　　fs.open(path, flags[, mode], callback)

　　参数使用说明如下：

　　　　path - 文件的路径。

　　　　flags - 文件打开的行为。

　　　　mode - 设置文件模式(权限)，文件创建默认权限为 0666(可读，可写)。

　　　　callback - 回调函数，带有两个参数如：callback(err, fd)。

1 fs.open('index.txt','r+',function(err,fd){

2 if(err) return err;

3 console.log(fd);

4 console.log('文件打开成功');

5

6 });

7、写入文件

　　fs.writeFile(filename, data[, options], callback)

　　如果文件存在，写入的内容会覆盖旧文件内容

　　　　参数使用说明如下：

　　　　　　path - 文件路径。

　　　　　　data - 要写入文件的数据，可以是 String(字符串) 或 Buffer(流) 对象。

　　　　　　options - 该参数是一个对象，包含 {encoding, mode, flag}。默认编码为 utf8, 模式为 0666 ，flag 为 'w'

　　　　　　callback - 回调函数，回调函数只包含错误信息参数(err)，在写入失败时返回。

IMG_261

1 var fs = require('fs');

2 console.log('准备写入文件');

3 fs.writeFile('input.txt', '我是新写入的内容', function (err) {

4 if (err) console.error(err);

5 console.log('数据写入的数据');

6 console.log('-------------------');

7 });

8 console.log('读取写入的数据');

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

10 if (err) console.error(err);

11 console.log('异步读取文件数据：' + data.toString());

12 })

IMG_262

8、读入文件

　　fs.writeFile(filename, data[, options], callback)

　　　　如果文件存在，写入的内容会覆盖旧文件内容

　　　　　　参数使用说明如下：

　　　　　　　　path - 文件路径。

　　　　　　　　data - 要写入文件的数据，可以是 String(字符串) 或 Buffer(流) 对象。

　　　　　　　　options - 该参数是一个对象，包含 {encoding, mode, flag}。默认编码为 utf8, 模式为 0666 ，flag 为 'w'

　　　　　　　　callback - 回调函数，回调函数只包含错误信息参数(err)，在写入失败时返回。

IMG_263

1 var fs = require('fs');

2 var buf = new Buffer(1024);

3 fs.open('input.txt', 'r+', function (err, fd) {

4 if (err) return console.error(err);

5 console.log('文件打开成功');

6 console.log('准备读取文件');

7 // fd fs.open的标识 8 // buf 缓存区 9 // 0, buf.length 缓存区区间10 // 0, 读取input.txt开始位置11 fs.read(fd, buf, 0, buf.length, 0, function (err, bytes) {

12 if (err) console.log(err);

13 console.log(bytes + ' 字节被读取');

14

15 if (bytes > 0) {

16 console.log(buf.slice(0, bytes).toString());

17 }

18 })

19 })

IMG_264

9、关闭文件

　　fs.close(fd, callback)

　　　　参数使用说明如下：

　　　　　　fd - 通过 fs.open() 方法返回的文件描述符。

　　　　　　callback - 回调函数，没有参数。

IMG_265

1 var fs = require('fs');

2 var buf = new Buffer(1024);

3 fs.open('input.txt', 'r+', function (err, fd) {

4 if (err) return console.error(err);

5 console.log('文件打开成功');

6 console.log('准备读取文件');

7 // fd fs.open的标识 8 // buf 缓存区 9 // 0, buf.length 缓存区区间10 // 0, 读取input.txt开始位置11 fs.read(fd, buf, 0, buf.length, 0, function (err, bytes) {

12 if (err) console.log(err);

13 console.log(bytes + ' 字节被读取');

14

15 if (bytes > 0) {

16 console.log(buf.slice(0, bytes).toString());

17 }

18 })

19 // 关闭文件 20 fs.close(fd, function (err){

21 if (err) console.error(err);

22 console.log('文件关闭成功');

23 });

24 })

IMG_266

10、截取文件

　　fs.ftruncate(fd, len, callback)

　　　　该方法使用了文件描述符来读取文件 参数

　　　　　　 fd - 通过 fs.open() 方法返回的文件描述符。

　　　　　　 len - 文件内容截取的长度。 callback - 回调函数，没有参数

IMG_267

1 var fs = require('fs');

2 var buf = new Buffer(1024);

3 console.log('准备打开文件');

4 fs.open('input.txt', 'r+', function (err, fd) {

5 if (err) return console.error(err);

6 console.log('文件打开成功');

7 console.log('截取10字节后的文件内容');

8 // 截取文件 9 fs.ftruncate(fd, 10, function (err) {

10 if (err) console.log(err);

11 console.log('文件截取成功');

12 console.log('读取相同的文件');

13 fs.read(fd, buf, 0, buf.length, 0, function (err, bytes) {

14 if (err) console.error(err);

15 // 仅仅输出读取的字节16 if (bytes > 0) {

17 console.log(buf.slice(0, bytes).toString());

18 }

19 // 关闭文件20 fs.close(fd, function (err) {

21 if (err) console.error(err);

22 console.log('文件关闭成功');

23 })

24 })

25 })

26 })

IMG_268

11、删除文件

　　fs.unlink(path, callback)

　　　　参数

　　　　　　path - 文件路径

　　　　　　callback - 回调函数，无参

1 var fs = require('fs');

2 console.log('准备删除文件');

3 fs.unlink('input.txt', function (err) {

4 if (err) return console.log(err);

5 console.log('文件删除成功');

6 })

12、创建目录

　　fs.mkdir(path[, mode], callback)

　　　　参数

　　　　　　 path - 文件路径

　　　　　　mode - 设置目录权限，默认为0777 callback - 回调函数

1 var fs = require('fs');

2 console.log('创建目录 test');

3 fs.mkdir('test', function (err) {

4 if (err) return console.error(err);

5 console.log('目录创建成功');

6 });

13、查看目录

　　fs.readdir(path, callback)

　　　　参数使用说明如下：

　　　　　　 path - 文件路径。

　　　　　　 callback - 回调函数，回调函数带有两个参数err, files，err 为错误信息，files 为 目录下的文件数组列表

IMG_269

1 var fs = require('fs');

2 console.log('查看 /file 目录');

3 fs.readdir('../file/', function (err, files) {

4 if (err) return console.log(err);

5 files.forEach(function (file) {

6 console.log(file);

7 })

8 })

IMG_270

14、删除目录

 　　fs.rmdir(path, callback)

　　　　 参数使用说明如下：

　　　　　　 path - 文件路径。

　　　　　　callback - 回调函数，没有参数。

IMG_271

1 var fs = require('fs');

2 console.log('删除 /test 目录');

3 fs.rmdir('test', function (err){

4 if (err) console.error(err);

5 console.log('读取 /test 目录');

6 fs.readdir('test', function (err, files) {

7 if (err) return console.log(err);

8 files.forEach(function (file) {

9 console.log(file);

10 })

11 })

12 });

IMG_272

 15、判断文件路径是否存在

　　fs.exists(path, callback)

**接收参数：**

　　　　　　path 欲检测的文件路径

　　　　　　callback 回调

1 fs.exists('/chengyi',function(exists){

2 if(exists){

3 console.log('路径存在');

4 }

5 });

16、向文件中追加写入

　　fs.appendFile(name,str,encode,callback);

　　　　参数

　　　　　　name : 文件名

　　　　　　str ： 添加的字段

　　　　　　encode ： 设置编码

　　　　　　callback ： 回调函数

IMG_273

1 fs.appendFile('index.txt','窗前明月光，疑是地上霜','utf8',function(err){

2 if(err) console.log(err);

3 fs.readFile('index.txt','utf8',function(err,data){

4 if(err) console.log(err);

5 console.log(data);

6 });

7 });

IMG_274

node 流操作  stream

    ---可读

    ---可写

    ---可读可写

    ---写入之后读出

事件：

    ---data    当数据可读时触发

    ---end     没有更多数据时触发

    ---error    在接收和写入的过程中发生错误

    ---finish    所有数据被写入到底层系统时触发（了解）

1.读取流



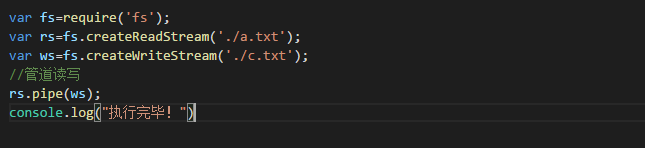
2.写入流



3.流的读取



4.管道读写



path 模块

    提供了一些工具函数 用来处理文件和目录的路径



服务器



