node.js服务器端执行js代码的方式。目的为了用js执行web服务器程序。

命令行模式/node交互模式

node --use\_strict xx.js开启严格模式（js文件内不需要’use strict’;）

安装node及其他模块的坑：

npm安装模块位置：应用程序所在目录的“node\_modules”文件夹，当前文件夹向上直到找到node\_modules文件夹。

修改express等模块默认安装位置：<http://www.jianshu.com/p/03a76b2e7e00>，这里有一个坑，即便设置好了也没有安到对应文件夹，此时要修改nodejs文件夹里modules下的npm里的npmrc文件，将代码prefix=${APPDATA}\npm更改为<http://blog.csdn.net/CareChere/article/details/51279789>所示内容。

express配置完毕后的使用：在项目文件夹下执行命令，比如

直接在localhost：3000查看输出内容。如果是静态文件，需要进一步配置，之后详细说明。

模块加载机制（CommonJS规范）

module.exports = 函数名 ;（或者{xx:xx,xxx:xxx}）

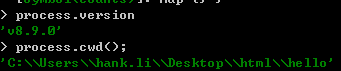
var greet = require(‘./xxx’) ;引入的上述模块作为变量保存在greet中。

基本模块

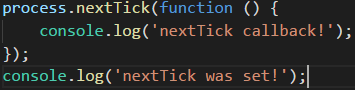
global node.js的全局对象（相当于浏览器中的window）

process 当前node.js的进程（global.process）

process.cwd() 当前工作目录



process.nextTick();在下次事件中响应（）内的代码（\*js是事件驱动的单线程模型）



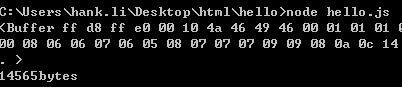
**fs 文件系统模块，读写文件**

var fs = require(‘fs’);

fs.*readFile*(‘xx.txt’,’utf-8’,function(err,data){

if(err){console.log(err);}else{console.log(data);})

上述为异步读取txt文件，也可读取图片（二进制文件），此时不需传入文件编码，data参数将返回一个Buffer对象，包含许多字节。注意，Buffer对象可以和String进行呢转换，如data.toString(‘utf-8’)，或者反之，Buffer.from(xxx,’utf-8’)。

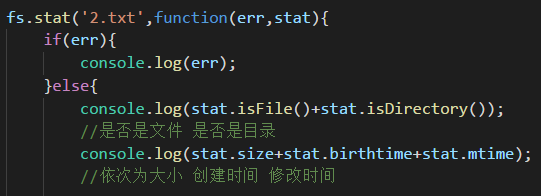


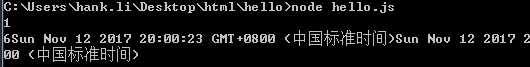
同步读取文件，则var data = fs.*readFileSync*(‘xxx.txt’,’utf-8’);。

fs.writeFile()写文件，参数依次为文件名、数据、回调函数。（写文件是覆盖原文件的）。同上，对应的同步方法为writeFileSync()。



fs.stat()获取文件大小、创建时间等信息，返回stat对象。





\*复制文件（同步读取实现） var source = fs.readFileSync(‘1.txt’,’utf-8’);

fs.writeFileSync(‘xx’,source);

该方法是把源文件内容放到内存，再将内存中的数据写入到目标文件中。大文件会撑爆内存。应读一些写一些，如此反复，即“流”（stream）。

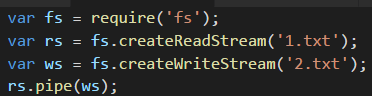
流分四种：readable writable duplex transform

读取数据的流继承自stream.readable；可以写入的流继承自stream.writable

var rs = fs.createReadStream(‘xx.txt’,’utf-8’);（写入为createWriteStream，调用write方法，最终以end()结束）

rs.on(‘data’,function(chunk){….});（流的事件有三种data（流可以读取）/end（流到末尾）/error（事件出错了））

pipe（将readable流和writable流串起来），流出流入，用于复制文件

注意，pipe()是readable中的方法。

**http协议**

参考：

<https://www.liaoxuefeng.com/wiki/0014316089557264a6b348958f449949df42a6d3a2e542c000/001432011939547478fd5482deb47b08716557cc99764e0000>

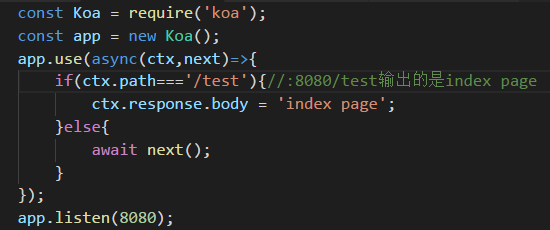
关键：

request：封装http请求

response：封装http响应，操作该对象的方法可以把http响应返回给浏览器

koa

ctx是koa传入的封装了request和response的变量



每个async函数称为一个middleware