NodeJS是运行在服务端的JavaScript的开发平台，实质是对chrome V8引擎的封装。

使用事件驱动和非阻塞I/O模型，使得其轻量，高效，非常适合在分布式设备上进行数据密集型的时时应用。

特点：

\*RESTful API

\*单线程：不需要额外的线程，依然可以对任务进行并发处理，原理是通过事件循环实现并发操作，因此应利用这一点，尽可能减少阻塞操作，多使用非阻塞操作

\*非阻塞I/O

\*V8虚拟机

\*事件驱动：用module模块划分不同的功能，简化开发，每个类库都有丰富的api，常见的模块有http，require，fs

vscode在当前目录打开命令行： 查看-Teminal 或 ctrl+反引号

vscode的插件node-snippets，输入node可根据提示自动创建代码结构

1、npm

npm是nodejs的包管理工具：

（1）初始化package.json：npm init

会生成package.json，内包含所需依赖的包

①npm install 会根据其内容安装依赖，别人只需这个文件就能自动装包，不需要发给其modules，所以git只发这个文件就行

npm init –y 全部选项直接yes

②npm update 更新包版本

③npm install --production npm install --dev 只安装生产环境/开发环境的包

（2）安装/卸载/包，清除缓存：

install可简写为I

npm install moduleName -g或—save --dev可选

-g或--global 全局安装

--save或-S 局部安装，生产环境，在开发/运行时都使用

--save --dev或-D 后只在开发时使用，运行时不适用

npm uninstall xxx –g/--save 卸载

清理npm缓存： npm cache clean –force 可解决包安装出现的一些问题

（3）查看已安装包：

npm list 或 npm list --ave 查看该项目的包和依赖关系

npm list | grep xxx 只显示xxx包的依赖关系

package-lock.jspn会显示已安装的包和依赖关系

npm list –g 查看全局包和依赖关系

npm outdated 查看已过期包

（4）node包的版本号：

①安装指定版本的包

npm install xxx@版本 指定版本

xxx@1 安装1.x的最高版本

@latest 最新版

npm view xxx versions 查看xxx的所有版本

②版本号：

主版本号.次版本号.补丁号

朱版本为大版本，一般有大更新

次版本号为小更新

补丁号偶数为稳定版本，奇数为不稳定版本

package.json的版本符号：a.b.c为例

^a.b.c 锁定a的版本，bc最新版本

~a.b.c 锁定ab的版本，c最新版本 （^的bc和~的c不要超过存在的版本，否则npm update报错）

a.b.c 锁定abc就是这个版本

\* abc都是最新版本

（5）导入包和运行js

引入模块：let xxxx=require(‘modulelName’)

注意：若引入内置/第三方包，不加./ 若引入自己目录下的包，必须./ 因为不加./会从module下找，加了./才会从自己目录找

运行js：cmd在js我那件的目录下，node xxx.js

（6）npm脚本：

package.json中

“script”:{

“xxx”:”ccc”,

}

给命令设置别名，执行npm run xxx 相当于 执行ccc

也可以自定义js文件 npm run xxx => node xxx.js

2、内置模块，第三方模块，自定义模块：

（1）内置模块

里面的回调函数的参数一般err在前，if(err)。。elsex 的形式，通常用来处理异常

①http

var http = require('http');

http.createServer(function (request, response) {

response.writeHead(200, {'Content-Type': 'text/plain'});

response.end('Hello World'); //必须有end才会结束响应

}).listen(8081);

request的api

url 请求的url

response的api

write(‘html结构’)

②url

url.parse(‘网址’) 解析网址的端口，域名，query等，传入第二个参数true能将query转对象

③path

④fs

//判断是目录还是文件

const fs = require('fs')

fs.stat('./123.txt',(err,data)=>{

if(err){

console.log('err')

return

}

else {

console.log(`文件？${data.isFile()}`)

console.log(`目录？${data.isDirectory()}`)

}

})

//创建文件

fs.mkdir('./666.css',(err)=>{})

//创建文件并写入内容，若文件已存在，则覆盖内容

fs.writeFile('./2.txt','123456789',(err)=>{})

//创建并写入内容，若文件已存在，则追加内容

fs.appendFile('...','...',(err)=>{})

//读取文件/目录内容，data为内容

fs.readFile('...',(err,data)=>{})

fs.readdir('...',(err,data)=>{})

//重命名，若路径不同，则是移动文件

fs.rename('...','...',(err)=>{})

//删除文件

fs.unlink('...',(err)=>{})

//删除目录，目录没有文件时才能删除目录

fs.rmdir('...',(err)=>{})

（2）常用第三方包：

lodash 字符层，数组，数学，集合，日期等api

debounce 防抖

（3）自定义模块：CommonJS模块化规范

自定义abc.js

module.exports = xxx

package.json中

“main”:”abc.js” 指定入口

2、express

基于nodeJS的简单灵活的web服务端框架，可以对数据库进行操作。

特性：

\*可以设置中间件来响应http请求

\*定义了路由表用于执行不同的http动作

\*可以向模板传递参数来动态渲染html

（）基本使用：

先建立一个存放服务器的文件夹，在该目录下：

npm install express --save

建立xxx.js文件，编写服务端代码：

const express=require('express')

//创建express对象

const app=express()

//创建路由规则

//GET请求时回调

app.get('/aaa',(request,response)=>{ //‘/aaa’是url

//设置允许跨域

response.setHeader('Access-Control-Allow-Origin','\*')

//响应体

response.send('123')

})

//POST请求时会带哦

app.post('/aaa',(request,response)=>{

//设置允许跨域

response.setHeader('Access-Control-Allow-Origin','\*')

//响应体

response.send('666')

})

//监听端口，一个端口只能有一个服务监听

app.listen(8000,()=>[

console.log('服务已启动') //启动服务器后的提示信息

])

与vue一样，可以配置动态路由：

get()和post()的url参数 ‘/xxx/:canshu’

参数request，response的属性/方法：

GET请求获取参数：

request.query.xxx 获取GET请求的query参数

response.params.canshu 获得动态路由的参数

POST请求获取参数：

npm install body-parser --save 普通参数

npm install multer --save cookie参数

npm install cookie-parser --save 文件上传的参数

安装后倒入，然后在post()中；

通过response.body 获取

启动服务器：

node xxx.js

request.query.xxx //获取GET请求的query参数

127.0.0.1:8000/url 可以调试

（2）自动重启服务器：

可以在修改，保存服务器js代码后自动重启

npm install nodemon --szve

启动服务器：npx nodemon xxx.js