一、什么是node

Nodejs：是一个异步的事件驱动的javascript运行时

C语言代码编译成可执行的文件的时候，这个时候算是编译时。C是需要编译才可执行的。而运行时环境就是相对于这个（不太明白）

二、Node.js的特性（其实是javascript的特性）：

1. 非阻塞I/O
2. 事件驱动
3. node与前端的js
   1. Js核心语法不变
   2. 前端主要是操作BOM和DOM
   3. 后端是操作fs（文件）、http（发请求）、buffer（操作二进制代码）、event（事件）、os（内存、操作系统）
4. Nodemon：保存项目时直接更新
5. Nodejs的模块包
   1. 核心模块（不需要require就可以用的）：
      1. Buffer：用于处理

Os模块：提供了与操作系统相关的使用应用程序方法和属性

Download-git-repo：下载git库

先npm i 安装，然后require引入

使用时download( github地址，存储下载的文件地址, 成功回调)

Ora：一个优雅的终端微调器 。



Util模块：使用工具

如：

1. util.promisify(original)

自定义的promise化函数

node流程控制问题：如何让异步任务串行化

一、方式

1. Promise

Const promise = name => new Promise( resolve => {

setTimeOut( () => {

Resolve()

}, delay)

})

1. Generator

Const generator = function \* (name) {

Yied promise(name);

Yield promise(name)

}

Const gen = generator(‘Generator’

Gen.next().value.then(() => {

Gen.next()

})

1. Async和Await组合ES7

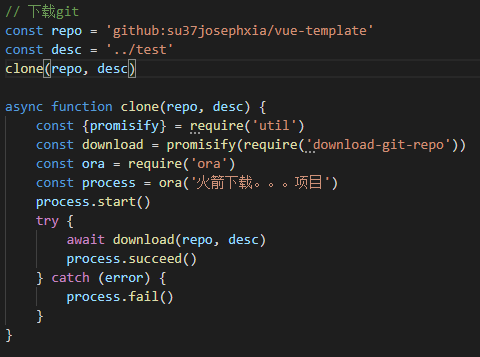
(async() => {

Await promise(‘Async/await’)

Await promise(‘Async/await’)

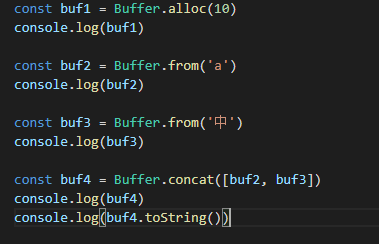
})()

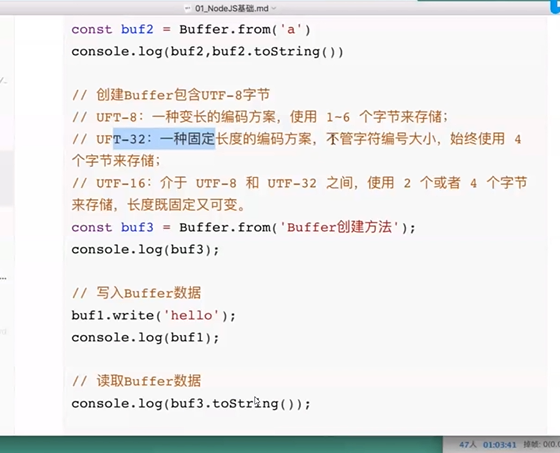
二、promisefy先让回调加工成promise



Buffer：千层封装的c语言的库，所有需要处理二进制文件的操作，尤其是读/写方面的，都需要Buffers

1. Buffer.alloc(‘10’) 分配一个10个字节的内存空间
2. Buffer.from(‘中’)





http: 服务相关

http.createServer：创建服务器

图片文件headers.accept中都会有’image/\*’这个字符串

fs模块：

1. 用于异步地读取文件的全部内容：

**fs.readFile(path[,options],callback)**

①、Path:目前使用过得是传递路径/目录，

②、options目前已知的是可以传encoding,即指定字符编码，如’utf8’。 如果没有指定encoding，则返回原始的buffer

③、fs.readFile()函数会缓冲整个文件，所以为了最小化内存成本，尽可能通过fs.createReadStream()进行流式传输

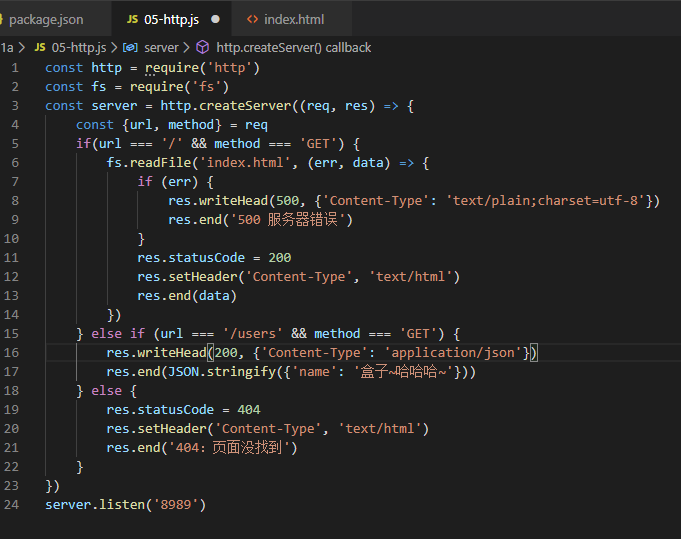
1. 从当前的文件位置开始顺序地读取

**fs.createReadStream(path[,options])**

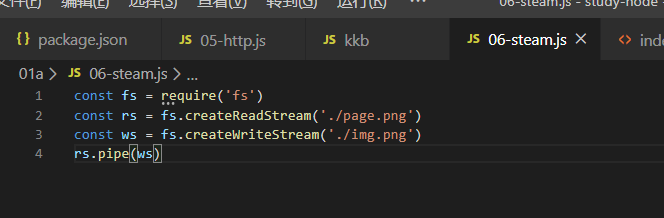
①、options中可以包括：encoding、start、end

1. 允许在文件的开头之后的某个位置写入数据

**fs.createWriteStream(path[,options])**



Stream流

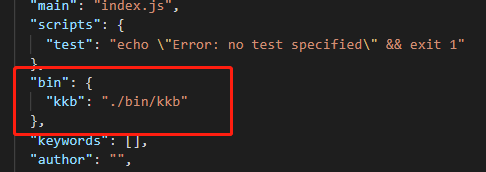


路由指引项目（类似于nuxt）

1. npm init -y：初始化项目，一路默认设置yes
2. 新建kkb文件（没有后缀。。。不明白）

#! /usr/bin/env node这里可以加注释，表明这是一个可执行的应用

1. 在初始化项目的package.json文件中配置
   1. package.json中，bin字段的用处：它可以用来存放一个可执行的文件



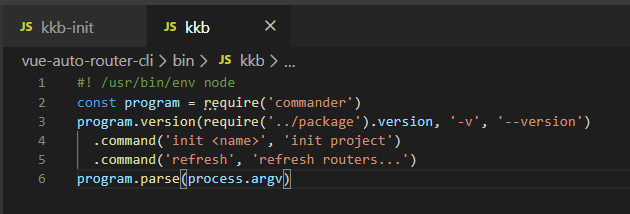
* 1. 接着，命令行中输入npm link创建全局命令。它将会把app这个字段复制到npm的全局模块安装文件夹node\_modules内，并创建符号链接，也就是将kkb的路径加入环境变量PATH

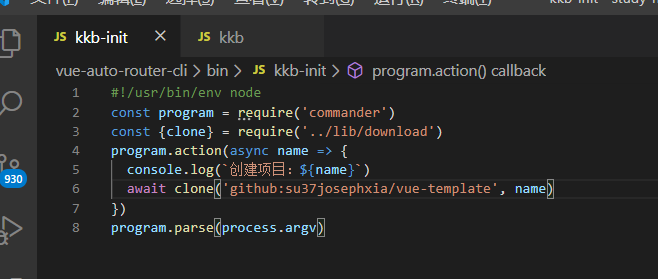
在命令行中输入kkb，即直接执行了kkb文件

1. 创建kkb-init文件，使用commander库去支持命令行操作

学习参考地址：<https://blog.csdn.net/phj_88/article/details/80944182>

1. 、command：定义命令行指令
2. 、alias：定义一个更短的命令行指令，还未成功实现过
3. 、description：描述，它会在help里面展示
4. 、option：定义参数，它接受四个参数，在第一个参数中，它可输入短名字 -a和长名字–app ,使用 | 或者,分隔，在命令行里使用时，这两个是等价的，区别是后者可以在程序里通过回调获取到；第二个为描述, 会在 help 信息里展示出来；第三个参数为回调函数，他接收的参数为一个string，有时候我们需要一个命令行创建多个模块，就需要一个回调来处理；第四个参数为默认值
5. Action：注册一个callback函数，这里需要注意目前回调不支持let声明变量
6. Parse：解析命令行





到此，会下载一个简单的vue项目sample,以下，开始实现约定路由

1. 创建kkb-refresh.js文件