一、nodejs基本介绍

1、 阻塞==》同步执行 非阻塞==》异步执行

2、在 Node.js 中 JavaScript 的执行是单线程的，因此并发性是指事件循环在完成其他工作后执行 JavaScript 回调函数的能力。任何预期以并行方式运行的代码必须让事件循环能够在非 JavaScript 操作（比如 I/O ）执行的同时继续运行。

3、事件：例如查询数据库结果可用或者用户点击了某个按钮都会触发事件。事件由事件处理程序或者事件回调函数进行 处理。

4、事件循环不同于许多其他语言的模型，其它语言创建额外线程来处理并发工作。

5、事件循环机制解析：

事件驱动编程风格和事件循环相伴而生，事件循环是一个处于不间断循环中的结构，该结构有两项功能：事件检测和事件触发处理。

在每一轮事件循环中，他都必须检测发生了什么事件。当事件发生时，事件循环还要决定调用那个回调函数。

主线程

客户代码（同步的）

异步请求1

其他线程

异步请求2

异步请求3

取事件

事件队列

执行回调

5、单线程：其实这里的“单线程”指的是我们（开发者）编写的代码只能运行在一个线程当中（可以称之为主线程吧），就像我们在Windows桌面程序开发中一样，编写的所有界面代码均运行在UI线程之中。

在nodejs中写的代码（包括回调方法）均只运行在一个线程中，但是不代表它只有一个线程。nodejs中许多异步方法在具体的实现时，内部均采用了多线程机制（具体后面会讲到）。

二、模块输出和引入：

1、commnd.js规范

2、定义模块：一个js文件就是一个模块；

3、输出模块：

module.exports={};

exports.method(or property)=\*\*\*\*;(注意这两种写法)；

4、引入模块：require()函数；

注意：（1）核心模块只能通过模块名引用；

（2）文件模块要写模块的相对路径。如果只写模块名称的话，则Node会依次在内置模块、全局模块和当前模块下查找hello.js

（3）文件夹模块：

var myModule = require(“./myModuleDir”)

Node会在指定的文件夹下查找模块。Node会假定该文件夹是一个包并试图查找包定义。包定义文件包含在名为package.json的文件中，main属性定义的文件。如果文件夹中不存在包定义文件package.json文件，则假定为默认值index.js

（4）如果模块名能被解析为相同的文件名，那么每次调用require(“myModule”)都会确切的返回同一模块。🡺只加载一次。

三、基本模块：

1、global对象：JavaScript有且仅有一个全局对象；浏览器中叫window对象，node中叫global对象。

2、process对象：也是Node.js提供的一个对象，它代表当前Node.js进程。

3、fs对象：文件系统模块，负责读写文件；和所有其它JavaScript模块不同的是，fs模块同时提供了异步和同步的方法。

fs.readFile(url,data,function(err,data){])//异步读取文件

fs.readFileSync(url,data);//同步读取文件；

fs.writeFile(url,data,function(err){});//异步写入文件；

fs.writeFileSync(url,data)//同步写入文件；

fs.stat(url,function(err,data){})//获取文件信息的异步函数；返回一个对象；

data.size;

data.birthtime;

data.mtime;

data.isFil();

data.isDirectory();

var sd=fs.stat(url);//同步获取文件信息；

4、1）tream是Node.js提供的又一个仅在服务区端可用的模块，目的是支持“流”这种数据结构。

rs=fs.createReadStream(url,'utf-8');

rs.on('data',function(chunk){}),rs.on('end',fn);rs.on('err',function(data));

ws=fs.createWriteStream(url,'utf-8');

fs.createWriteStream(new Buffer('str','utf-8'));

pipe();连接两个流；rs.pipe(ws);

默认情况下，当rs流的数据读取完毕，end事件触发后，将自动关闭ws流。如果我们不希望自动关闭ws流，需要传入参数：rs.pipe(ws, { end: false }

2） Stream 有四种流类型：Readable - 可读操作。Writable - 可写操作。Duplex - 可读可写操作。Transform - 操作被写入数据，然后读出结果。

3）所有的 Stream 对象都是 EventEmitter 的实例。常用的事件有：data - 当有数据可读时触发。end - 没有更多的数据可读时触发。error - 在接收和写入过程中发生错误时触发。finish - 所有数据已被写入到底层系统时触发。

5、http对象：request对象封装了HTTP请求，我们调用request对象的属性和方法就可以拿到所有HTTP请求的信息；

6、response对象封装了HTTP响应，我们操作response对象的方法，就可以把HTTP响应返回给浏览器。

7、crypto对象: crypto模块的目的是为了提供通用的加密和哈希算法。

8、模块系统：

四、npm

1、npm 用来管理node包。不仅会安装你指定的包，还会安装这些指定包所依赖的其他包。比如你需要安装A包，而A包又依赖B和C，则node会将B和C安装到./node\_modules/A/node\_modules中

2、常见命令

npm install <package name> 安装包到本地

npm install <package name> -g 安装包到全局

npm install <package name>@2.0.5 安装指定版本

npm install <package name>@2.0.x 安装版本号前缀为2.0的最新的包

npm install <package name>@”<0.3”安装版本号小于0.3的最新的版本包

npm install [sax@”>=0.1.0](mailto:sax@) < 0.3.1”安装版本号大于等于0.1.0小于0.3.1的最新的版本包

npm uninstall <package name>卸载本地版本包

npm uninstall <package name>卸载全局安装的版本包

npm update <package name>更新版本包

五、缓冲区处理、编码和解码二进制数据

1、Buffer类：Buffer类的数据占用的内存不是保存在js VM中，而是占据一个不会被修改的永久内存地址。避免了缓冲区内存赋值所造成的cpu浪费。

六、事件发射器

1、var em = new (require(‘event’).EventEmmiter)();

var listenerFun = function() {

console.log(“hello”)

}

em.on(“hello”, ())

em 事件发射器

on 注册事件

hello 事件

listenerFun 事件监听器

2、removeListener(“事件”，事件监听器)

RemoveAllListener(“事件”)

addEventListener(“事件”，事件监听器)

on(“事件”，事件监听器)

once(“事件”，事件监听器)===只执行一次：如em.emit(“事件”) em.emit(“事件”)只会执行一次注册的事件。