Nodejs背景+基础知识+基础操作

Git地址：<https://github.com/U8Learner/NodeJs>

后边的asp.net mvc项目也会放在U8Learner下，后续讲完了，个人的新增项目也会放在这个下边。

Git上建了组织U8Learner,有兴趣的同学可以加入组织，一起学习。

只讲干的（技能），不讲湿的（思想，流程），时间不够，世界上最困难的事情就是改变人的思想。

1. 什么是nodejs?

Node 是一个服务器端 JavaScript 解释器，它将改变服务器应该如何工作的概念。它的目标是帮助程序员构建高度可伸缩的应用程序，编写能够处理数万条同时连接到一个（只有一个）物理机的连接代码。高并发能力！

特点：非阻塞、事件驱动

简单理解：以前js只能在浏览器里跑，一个老外用了chrome最新的v8引擎（高效）写了node的内核和相关库，让js在非浏览器中也可以跑起来，也可以作为服务端的开发语言，尤其是适用于高并发的场景，这是node的特点决定的。

1. 为什么要用js?为什么要用nodejs?

Js:web不用js客户端怎么写？方便，什么都不需要，一个浏览器就可以工作。更接近客户，更接近体验端，让客户端多干事儿，服务端少干事儿，减少对服务端资源的消耗。

Nodejs:客户端服务端同一种语言，性能高，尤其适用于高并发场景。前后端分工比较明确的公司，你和前端开发同学说，你本地调试启动个jetty/tomcat/IIS，前端同学会和你拼命。对前端同学也是不合理的，每次后端依赖文件出问题，编译不通过，前端同学就干不了活儿。安装nodejs的同时会安装npm工具，npm(node package management),.net->nugget,java->maven

Js->npm，没有依赖管理，开发效率不可能高。团队项目源码托管的要求，我认为最基本的要求是：拉下代码，执行简单命令，代码立马就能跑起来，而不是到处去配置和查找依赖包。

社区活跃，开源项目多，混搭可选择的多，开源代码质量一般不低。

1. 安装nodejs

<https://nodejs.org/en/> 选择合适自己机器（win/linux/mac）的版本下载

安装后分别运行命令

node –v

nmp –v

查看是否正确输出版本号

1. 安装nosql(mongodb)

Windows 用户向导：<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>

Linux 用户向导：<https://docs.mongodb.com/manual/administration/install-on-linux/>

Mac 用户向导：<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/>

获取自己去博客园查找资料，很多。

Mongodb界面工具：robbomongo(跨平台，开源，免费，小巧)

下载地址：<https://robomongo.org/download>

monogoChef（另外一种界面工具，收费，大，功能全）

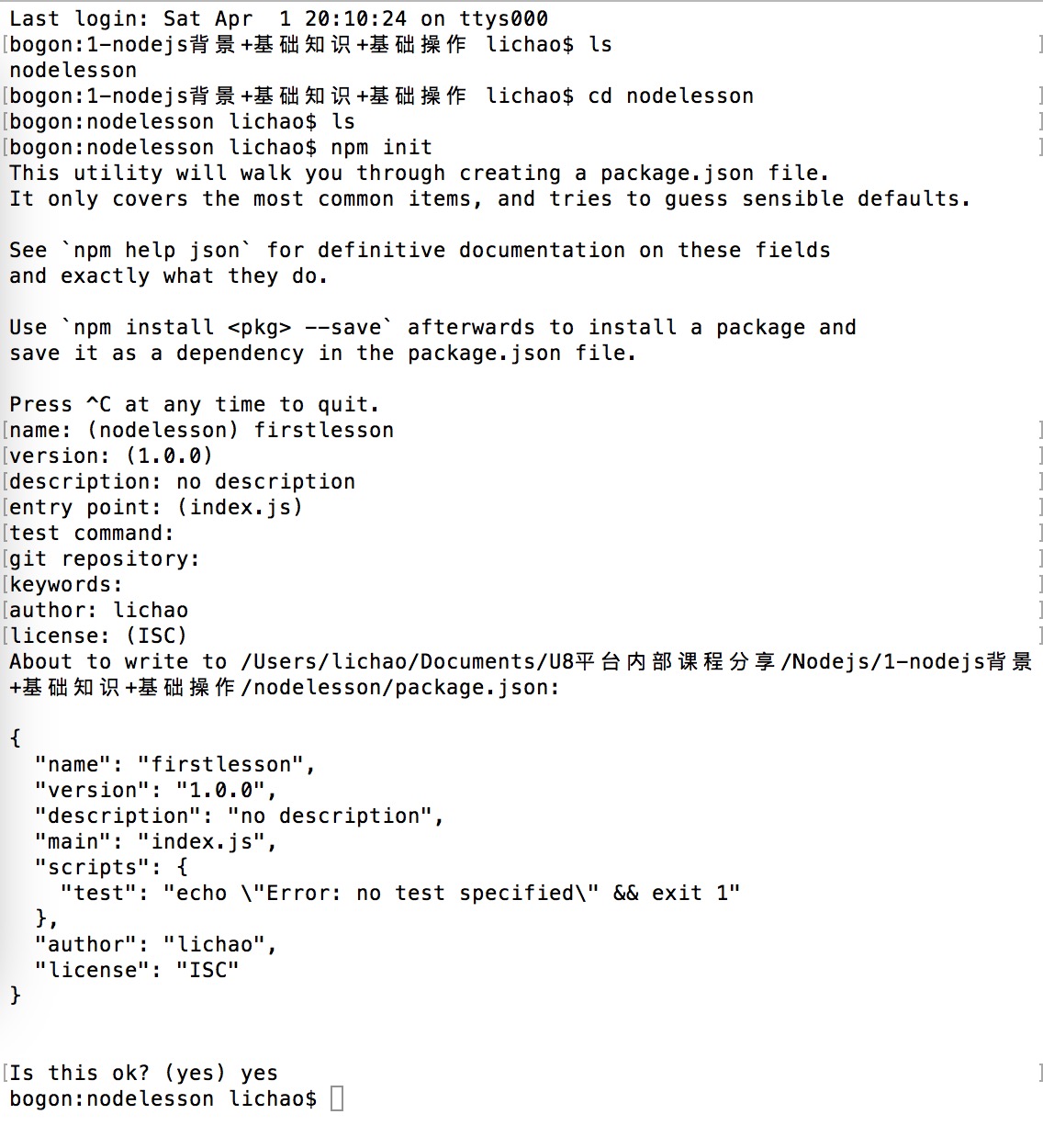
<http://3t.io/mongochef/#mongochef-download-compare>

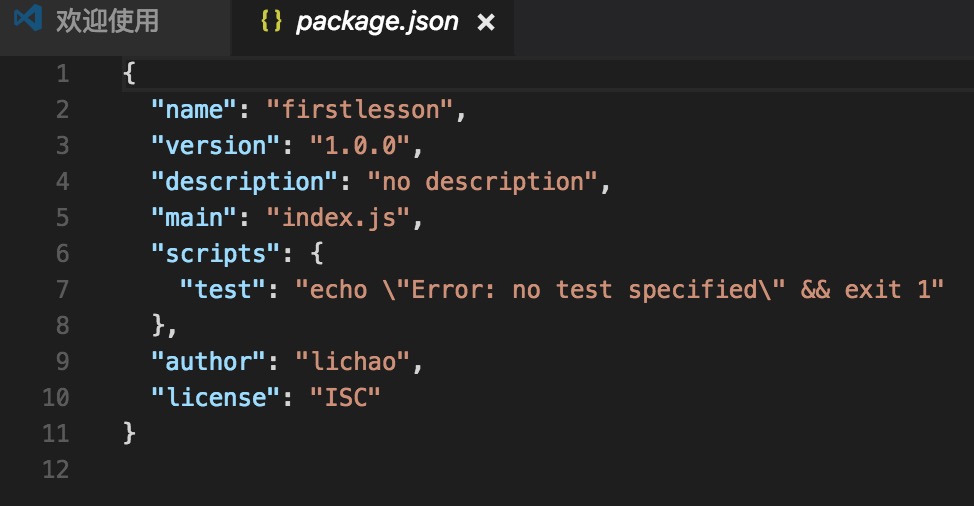
1. 开发工具

推介vs code（跨平台，开源，免费，插件模式，各种语言支持，未来如果对.net core感兴趣的更应该用）

1. nodejs开发的简单使用

安装好node环境后，在任意位置新建文件夹nodelesson





package.json 以json格式存储的对当前项目的配置信息

使用第三方包，做一个简单的express应用：

npm install express npm install express –save

安装完成后，项目下会多一个文件夹node\_modules,下边全是你安装过的第三方依赖包。加—save代表你要将安装信息保存到package.json这个文件中，这样，别人使用你的项目时候，git上来下来代码，然后npm install,就可以直接查找配置文件的信息，得到所有依赖的第三方包，然后下载到本地，无需自己配置依赖，立即可以运行。

注：安装第三方包是递归依赖的，也就是如果你安装的第三方包也依赖其他第三方包，npm会自动帮你下载，并且放到node\_modules文件夹下

新建app.js文件，在文件中写入：

// 这句的意思就是引入 `express` 模块，并将它赋予 `express` 这个变量等待使用。

var express = require('express');

// 调用 express 实例，它是一个函数，不带参数调用时，会返回一个 express 实例，将这个变量赋予 app 变量。

var app = express();

// app 本身有很多方法，其中包括最常用的 get、post、put/patch、delete，在这里我们调用其中的 get 方法，为我们的 `/` 路径指定一个 handler 函数。

// 这个 handler 函数会接收 req 和 res 两个对象，他们分别是请求的 request 和 response。

// request 中包含了浏览器传来的各种信息，比如 query 啊，body 啊，headers 啊之类的，都可以通过 req 对象访问到。

// res 对象，我们一般不从里面取信息，而是通过它来定制我们向浏览器输出的信息，比如 header 信息，比如想要向浏览器输出的内容。这里我们调用了它的 #send 方法，向浏览器输出一个字符串。

app.get('/', function (req, res) {

res.send('Hello World');

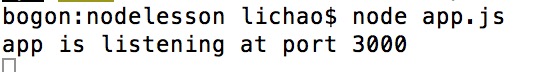
});

// 定义好我们 app 的行为之后，让它监听本地的 3000 端口。这里的第二个函数是个回调函数，会在 listen 动作成功后执行，我们这里执行了一个命令行输出操作，告诉我们监听动作已完成。

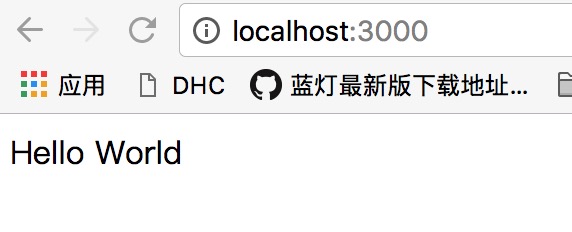
app.listen(3000, function () {

console.log('app is listening at port 3000');

});



终端运行node app.js，提示服务器已经在3000端口监听，用浏览器访问这个地址。



1. js的调试：

主流：chrome->f12->console/network……..

软流：f5（使用node时候，文件，文件夹尽量全部用英文，用中文，出了问题找不出原因。。。），这种方式下，如果用中文，有时候会导致断点进不去，打了断点没反应。

顺便：主流开发（普适和开源） 软流（一家之言，发展好坏取决大部分开发者的态度） ts和js举例