## 模块系统

模块系统中如果需要导出 则使用exports.xxx进行导出

使用 foo.xxx进行访问

如果只需要导出一个变量或者对象的话 则使用 module.exports = xxx

直接使用foo进行访问

// 个人理解 nodejs的文件作用域(模块系统)在文件开头自动执行

exports = module.exports

// 具体下来就是一个引用类型  exports和module.exports指向同一个内存地址

// 所以可以在文件当中使用

exports.xxx = "xxx"

// 在文件最后自动执行

return module.exports

npm中的包加载规则

从本目录开始层层向上查找node\_modules目录(如果本目录下买有main属性标识的文件,也会层层向上查找,如果没有找到,则报 can not find module "xx"错误)

找到node\_modules中的xx文件夹

xx文件夹-> package.json-> main属性(默认为index)

### 包说明文件

-- save 写入包描述文件,生产依赖 可以省略 缩写 -S

-- save –dev 开发依赖 缩写 –D

升级npm

npm i npm –g

### npm常用命令

npm init 生成package.json 使用npm init –y可以跳过向导快速生成

npm install 根据package.json下载依赖

npm install 包名 只下载包

npm install –save 下载并且保存依赖项 简写 npm install –S

npm uninstall 包名 删除包,只会删除被删除的,但是它的依赖项会保留下来

npm unistall –-save 包名 删除的同时也会把依赖信息去除 简写 npm un –S 包名

npm help 查看使用帮助

npm 命令 –-help 查看指定命令的帮助

npm config list 查看npm配置信息

## express

原生的http在某些方面表现不足以应对我们的开发需求,所以我们就需要使用框架来加快我们的开发效率,框架的目的就是提高效率,让我们的代码高度统一

在Node中有很多web开发框架,现在以学习express为主

### 使用

1. 安装

npm i express

1. 引入

var express = require("express")

1. 创建服务器应用程序 // 也就是原来的http.creatServer

// 调用express  
**var** app = express()  
// 公开 public文件夹  
app.use("/public/",express.static("./public/"))  
// 请求为/时返回  
app.get("/",**function** (req,res) {  
 res.send("<h1>你好,express<br> <a href='/about?a=1&b=2'>关于我</a> </h1>")  
})  
// 请求为about时返回  
app.get("/about",**function** (req,res) {  
 console.log(req.query)  
 res.send(`<h3>关于我,express,req:${req.url}</h3>`)  
})  
// 开始监听  
app.listen(3000,**function** () {  
 console.log("服务启动")  
  
})

配置post请求body请求体

app.use(bodyparser.urlencoded({ extende: **true** }));  
app.use(bodyparser.json())