**01 APP开发简介**

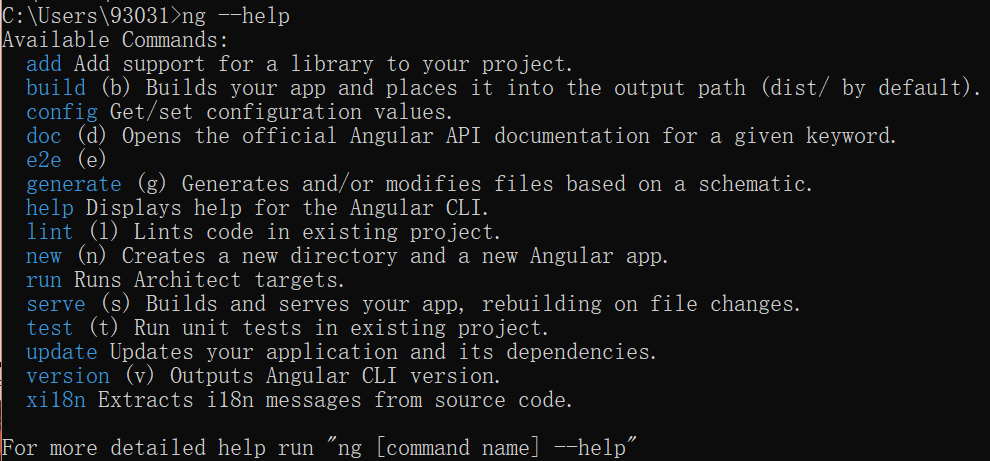
1. APP开发模式
2. 混合应用开发框架介绍

**TypeScript**【是个真正的编程语言】是JavaScript的超集，兼容JavaScript，扩展名为.ts

**02 Angular基础**

1. 环境搭建
2. 项目创建及概览
3. 模板语法
4. 创建项目：ng new 项目名称

进入项目：cd 项目名称

启动服务器：ng serve –-open



【可以看help后跟的功能】

1. Module模板 component组件
2. 项目启动过程：augular.json定义应用程序 启动时加载的HTML页面（index.html）和脚本文件（main.ts）
3. a. 插值表达式：

.ts

export class AppComponent{ //这是一个类

//报错a:number = ‘100’;【赋值错误，检测能 力比较强】

title:string = ‘hello’;

}

.html

<h1>{{title}}</h1> 【插值表达式】比较简单

【直接获取title，网页显示hello】

b. 属性绑定：

<img [src] = “imgUrl”> or <h3 [innerHTML]=” ‘hello’ ”></h3>【看清双引号中还有单引号，表示字符串，若不加则为变量】->显示hello

c. 事件绑定：

<button (click)=”fun()”>点击</button>

export class AppComponent{ //这是一个类

//报错a:number = ‘100’;【赋值错误，检测能力比较强】

title:string = ‘hello’;

fun(){

//报错：title = ‘hello angular’;【类中访问自身的对象，需要用this.访问】

this.title = ‘hello angular’;

}

}

双向数据绑定：【input标签最有用】【属性绑定：组件->模板 事件绑定：模板->组件】

[(ngModel)] 引入：FormsModule(在app.module.ts)

import { FormsModule } from '@angular/forms';

imports: [

BrowserModule,

FormsModule

],

<h3 [innerHTML]=”txt”></h3>

<input type=”text” [(ngModel)]=”txt”>

export class AppComponent{ //这是一个类

//报错a:number = ‘100’;【赋值错误，检测能力比较强】

title:string = ‘hello’;

fun(){

//报错：title = ‘hello angular’;【类中访问自身的对象，需要用this.访问】

this.title = ‘hello angular’;

}

txt:string;

}

1. 条件判断 \*nglf 【可以在DOM中添加或者移除这个元素，若表达式为真，则有这个DOM元素，反之移除（移除：在网页中没有，但是控制台只是注释掉了）】

<button (click)=”fun()” \*nglf=”20>10”>点击</button>【有个按钮，nglf里可以跟个布尔表达式】

1. export class AppComponent{ //这是一个类

//报错a:number = ‘100’;【赋值错误，检测能力比较强】

title:string = ‘hello’;

fun(){

//报错：title = ‘hello angular’;【类中访问自身的对象，需要用this.访问】

this.title = ‘hello angular’;

}

txt:string;

arr:number[]= [1,2,3];

}

<ul>

<li \*ngFor=”let item of arr”>

这是第{{item}}个

</li>

</ul>

含索引循环<div \*ngFor="let item of arr;let i=index">{{item}}</div>【index是下标】

<button (keydown)=”fun($even)”>点击</button>

<ul>

<li \*ngFor=”let item of arr;let i=index;”>

<input type=”checkbox”>这是第{{item}}个，下标是{{i}}

<button (click)=”del(i)”>删除</button>

</li>

</ul>

export class AppComponent{ //这是一个类

//报错a:number = ‘100’;【赋值错误，检测能力比较强】

title:string = ‘hello’;

fun(e){

//报错：title = ‘hello angular’;【类中访问自身的对象，需要用this.访问】

this.title = ‘hello angular’;

if(e.keyCode==13){

//报错this.arr.push(this.txt)应该数据统一

}

}

txt:string;

arr:number[]= [1,2,3];

//del(idx){

//console.log(idx);

//}

del(idx :number){

this.arr.splice(idx,1);

}

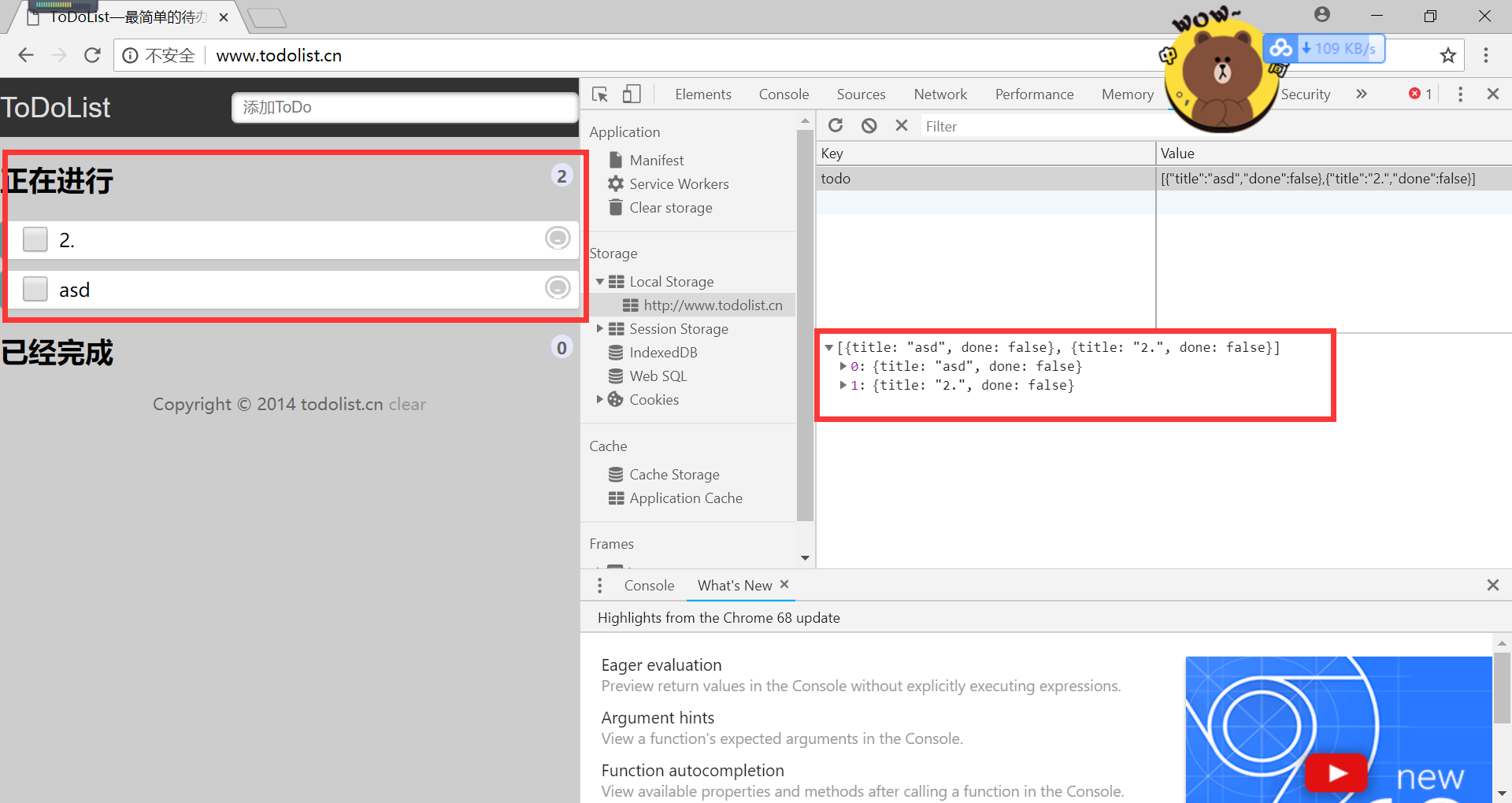
}

作业：【完成todolist】 【D:/angular/angulardemo】

注意：

（1）回车事件：(keydown)="fun($event)" ($event)事件对象

Todolist网页中，之前的数据刷新后依然存在，不像我们写的网页刷新一下又是新的



利用数组存储

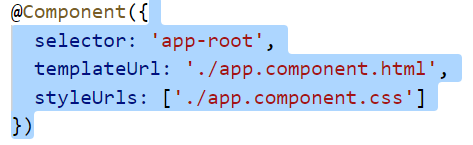
（2）Angular模板循环 ng-container：

ng-container相当于if语句 不想显示的时候使用这个标签

**03 angular组件**

1. 框架模式
2. 组件
3. 生命周期
4. 组件：组件负责控制屏幕上的一块区域，即视图

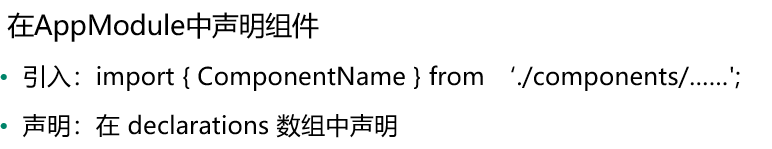
每个Angular应用至少有一个NgModuel类【模块比组件高级，根模块总会有个跟组件】

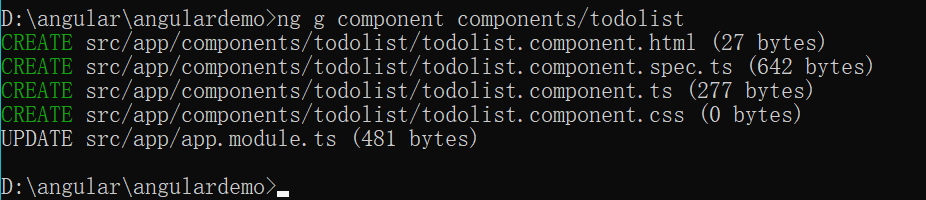


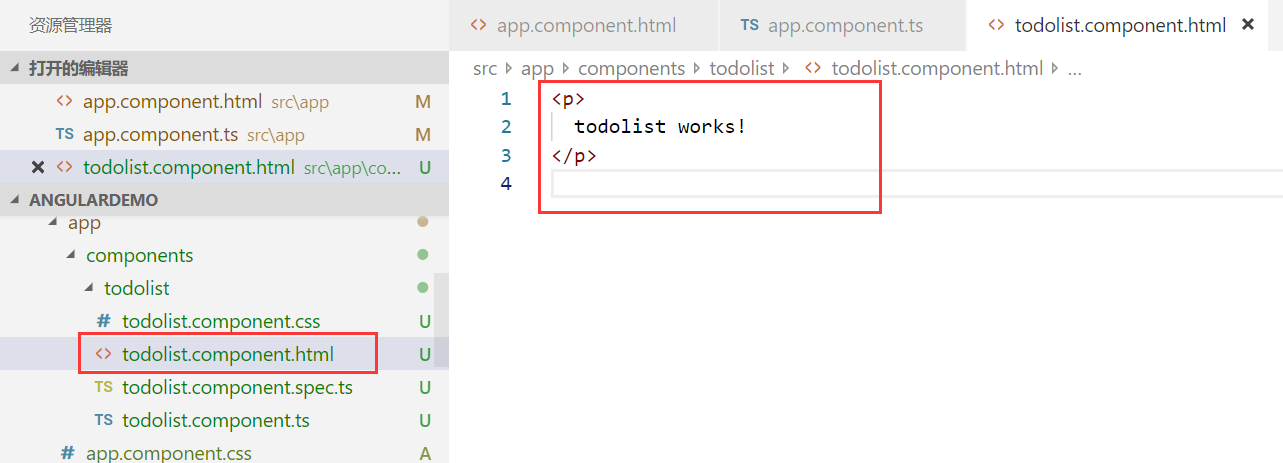
创建组件：ng g component components/componentName

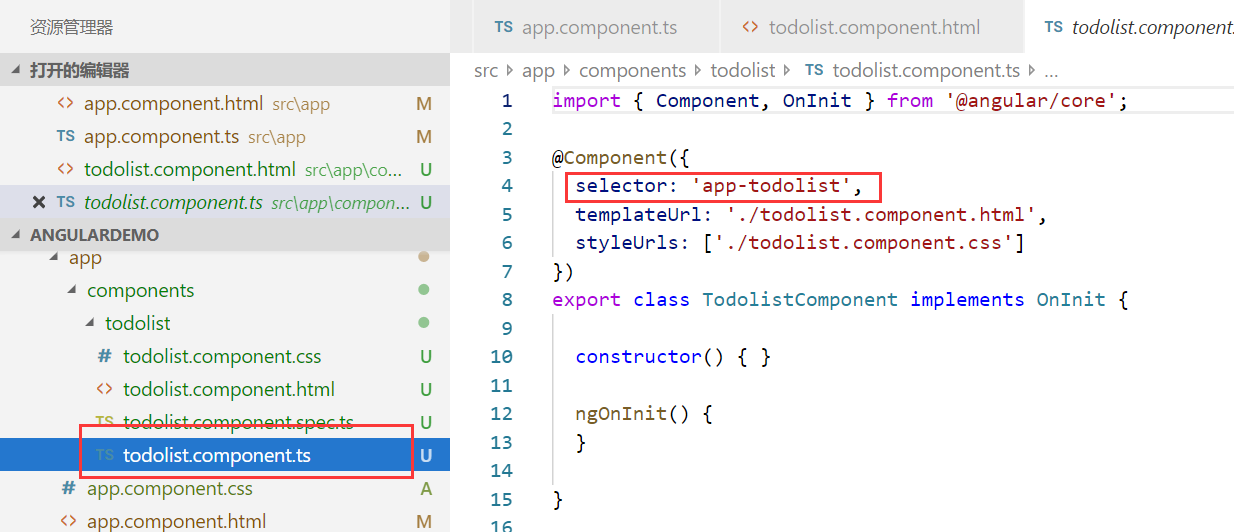
【如，ng g component components/todolist】 g 为创建的意思

【最好通过命令创建组件，angular会自动添加申明，如果手动创建，还得引入和申明】









使用组件：<app-组件名></app-组件名>



1. 组件交互 input、output【一定看angulardemo】

作业：把todolist分成两个组件，输入框一个 列表一个 数据需要一致

1. 生命周期

生命周期钩子

1. ngOnChanges()【检测到组件输入属性发生变化时调用 引入：在核心模块中加SimpleChanges】
2. ……

【复习】

1. 构造函数、原型
2. 数据类型
3. DOM节点操作、属性
4. 页面优化、页面回流、页面重绘
5. 正则表达式【匹配大括号{{name}}…】
6. 文档碎片
7. 解构【赋值】
8. 递归函数
9. 数组、字符串的方法
10. Object.keys()
11. 浮动原理
12. 定位的原理
13. BFC、IFC
14. 边距重合
15. 高度塌陷

**04 angular服务**

1. 服务创建：打开angular项目，然后ng g service services/serviceName
2. 字符串转化成数组：slipe(“,”) 【例，this.data=this.local.get('list').split(",");】

复习：模块【module】、组件【component】、服务【service】、依赖注入

1. 依赖注入：当应用程序的一些部件需要另一些部件时，利用依赖注入来创建被请求的部件，并将它们注入到发出请求的部件中【想使用什么服务（其实是一个类，类中各种属性和方法），依赖这个类】
2. 控制反转：
3. 控制反转是我们最终想要实现的目的，依赖注入是实现控制反转的手段
4. 注入器（Injector）：提供一系列接口用于创建依赖对象的实例【这个是不用开发者操作的，一般操作的都是提供框架的人员，自己的项目不用操作注入器】
5. 提供器：配置注入器【注入器有什么服务，提供器就通过注入器来创建被依赖对象的实例】



Provide保存的是providers的唯一标识，用于定位依赖值和注册提供器

Useclass指定创建实例的服务类

1. 依赖：只用申明依赖对象，注入器会根据这个依赖对象的类型创建对应的对象【这个过程，代码中是没有的】
2. 层级注入：注入器 -> 应用级注入 -> 主组件 -> 子组件
3. 限定方式注入【用得不多】

@Optional( )：兼容依赖不存在的情况，提高系统健壮性【若没有这个服务，还有页面显示，不适用服务的方法，就不报错】

@Host( )：限定查找规则只停止于当前组件【无此服务，直接瘫痪，无法使用】

1. 可以通过服务建立两个组件之间的关系

**05 路由**

1. 路由的配置：index.html中head标签下添加base元素【一般不用设置，系统自己配置】
2. 路由出口 <router-outlet></router-outlet>

路由链接 a [routerLink]= [“/”] [routerLinkActive]=[“”] ></a>

routerLink：绑定导航的路径

1. 路由传参：<a [routerLink]= [“/”] [queryParams]=“{属性：属性值}” ></a>



如果多个参数，则用&分开

1. JS跳转【引入router模块，构造函数中声明】

Router.navigate()方法 路径传参 查询参数传参

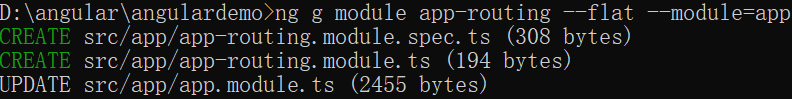
1. 父子路由

路由组件被嵌入另一个路由组件，建立多级嵌套关系

{ path : “路径”, component :“组件名”, children :[

{ path: “路径”, component:“组件名”}，……

]}



封装bind函数【双向数据绑定】 【bind.html】

Bootstrap

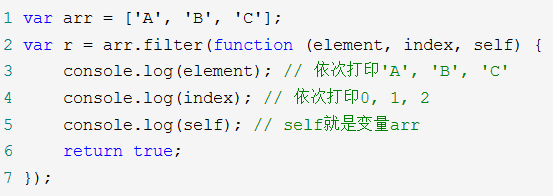
**06 HTTP服务**

1. // ng server --open --proxy-config proxy.conf.json

启动服务=>任务中npm run start

【"start": "ng serve --open --proxy-config proxy.conf.json"】

1. **filter()【过滤器】**接收的回调函数，其实可以有多个参数。通常我们仅使用第一个参数，表示Array的某个元素。回调函数还可以接收另外两个参数，表示元素的位置和数组本身



1. get请求 post请求

//body的解析在server.ts import bodyParser from 'body-parser'; const app = express(); app.use(bodyParser.json());

get请求：

this.http.get('/api/courses/3').subscribe((data)=>{ //调用请求函数

post请求：

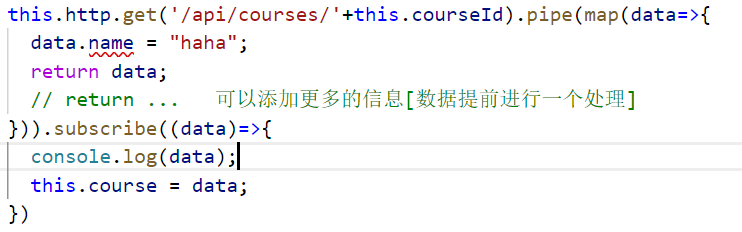
this.http.post('/api',{name:"xixixi"}).subscribe((data)=>{

**//HttpClient.post(url: string, body: any, options)**

1. post中body的解析

body的解析在server.ts import bodyParser from 'body-parser'; const app = express(); app.use(bodyParser.json());

1. headers
2. RxJs 在项目中 import { filter, map } from 'rxjs/operators';



【报错不用管，因为最开始没有name这个属性】



启动两个服务 server: nodemon serve.js 项目里: npm run start

**07 Angular指令和管道**

1. 指令

作用在特定的DOM元素，扩展该元素的功能，为元素增加新的行为

1. 指令的分类：

（1） 属性指令

（2） 结构指令

（3） 组件

（4） 内置指令【angular内置的】=>通用 路由 表单

通用

**ngClass**同时添加或移除多个CSS类

**[ngClass] = “{}”** [class.active]=”true”;【会显示css中.active的样式】

[class.active]=”3<7”;【也可以直接写表达式(布尔值)=>显示样式或不显示】

[ngClass]=”{‘active’:true}”

**ngStyle** 同时设置多个内联样式

<li [style.color]=” ’red’ ”> </li>【控制一个的】

<li [ngStyle]=”{ ‘color‘:’red’,’font-size’:’15px’}”【控制多个】

**ngSwitch** 类似switch语句，可以从多个可能的元素中根据switch条件来显示某一个【与\*ngSwitchCase、\*ngSwitchDefault连用】

**【注意引号个数】**

1. 管道

<p>{{jsonObject | json}}</p>

<p>{{date | date:'y-MM-dd'}}</p>

{{'fengjiadong' | uppercase}}



异步管道：<h3>{{(course|async)?.id}}</h3>

1. 自定义管道：ng g pipe pipes/pipeName

图标：<http://www.iconfont.cn/>

搜索图标 -> 添加图标 -> 复制代码（下载到本地）

在css中添加样式：【复制的代码】

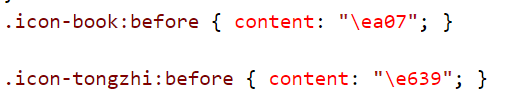


在所用的页面（.html）



**方法二：**

在下载代码的iconfont.css文件中



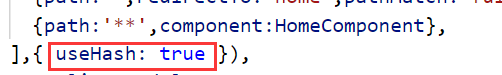
添加到css中，再在html中



打包：

在项目中ng build【自动打包到项目下的dist文件夹中】，复制该项目到server下，启动nodemon server.js就行，网址上直接输入请求端口号【例如，localhost:8080/】

刷新的时候找不到页面 cant Get… => 在app.module.ts路由的设置中



**Ionic**

**01 Ionic基础**

1. 3.9.2版本 基于angular

第四版 可以基于其他

1. <https://github.com/ionic-team/ionic-cli>
2. 环境搭建npm install -g [ionic@3.20.0](mailto:ionic@3.20.0)
3. Ionic -v 版本号的查询
4. Ionic出现Ionic的命令，关于Ionic的信息
5. 框架 cordova 是ionic里的一个插件 npm install [cordova@8.0.0](mailto:cordova@8.0.0) -g
6. Ionic项目创建 ionic start 项目名称 tabs【参数】

如果项目创建失败 进入项目 npm install【就继续下载刚才没有下载完的文件】

1. 清除缓存 npm cache clean –force