### 一．Ng2

Day1 ················································1

Day2 ················································1

Day3 ················································1

Day4 ················································1

Day5 ················································1

Day6 ················································1

Day1

一、AngularJS概述

官网：

https://angular.cn/ 中文

https://angular.io/ 英文

哪些公司在使用AngularJS？

Google、网页版微信、锤子手机。。。

what?

使用的angular是哪个版本？ ng4.0 与老版本的区别

①ng2.0之后所有的版本 基本上 没有大的变化

②ng1.\* 与 ng2.0之后所有版本的区别 就像是雷锋和雷峰塔的关系

③命名：将angular1.\* 称之为angularJS,将angular2.\*以及以后所有的版本统称为angular

④angular引入了component的概念，舍弃了老版本的$scope/controller..

⑤angular速度更快

⑥angular的代码由于是基于组件化的开发，代码的可读性非常好，调试方便，测试简单

angular是一个Google推出的js的框架，

when?

主要实现SPA（single page application 单页面应用程序），用在数据操作比较频繁的场景；是实现大而全的平台，能够实现移动端和桌面端的开发

why?

angularJS在一些传统企业中依然在大量使用，angular主要是用在新兴的互联网公司。

①迎合开发的潮流，引入了组件的概念

（提高了代码的复用率，降低了测试难度，提高了开发速度）

②代码具有更强的可读性和可维护性

③angular引入了更优秀的提高性能的工具，让开发和部署更简单，让应用程序速度更快

④angular采用的语言是 typescript

ts是由微软和google合作联合推出的一种基于es6的超集的语言；支持了新特性，比如：js是弱类型检查语言，而ts是强类型检查语言，让代码更健壮！

how?

如何来搭建开发环境？

提供两种方式，基于cli：

方式1：官方推荐

Angular CLI是一个命令行界面工具，它可以创建项目、添加文件以及执行一大堆开发任务，比如测试、打包和发布。

①安装nodejs

如果node -v ; npm -v能够显示对应的版本信息，已经安装，

否则nodejs.org官网下载最新版本，一路next去安装

②通过npm安装 angular cli

npm install -g @angular/cli

③通过安装的cli去创建一个项目

ng new my-app

④启动开发服务器

cd my-app

ng serve --open

方式2：

借助于angular所提供的quickstart

(17,819 个文件，2,434 个文件夹)

①下载快速开始用到的文件

https://codeload.github.com/angular/quickstart/zip/master

②解压缩

安装该项目依赖的所有的资源文件

npm install

（将nodeModules.zip文件解压缩，将拷贝到quickstart-master文件夹中）

③启动开发服务器：

npm start

二、AngularJS开发基础知识

1、执行npm start时，Angular是如何启动？

在启动开发服务器的时候，会去寻找src目录中main.ts文件；

在main.ts文件中，指定要启动一个模块AppModule；

在src目录中有一个app目录，有一个app.module.ts文件；

在app.module.ts文件中，指定bootStrap要启动哪一个组件？--》 AppComponent（一般情况下，都是整个应用程序的根组件）

在app.component.ts文件中，指定了template用来渲染该组件的视图（Hello Angular）

2、组件的创建和使用

创建：

在app目录中，创建一个文件夹，在该文件夹中创建一个\*\*\*.component.ts(比如：demo01.component.ts)

在该文件中， 通过 @Component装饰当前的类为一个组件类，然后通过export导出一个类

使用：

①先到模块中声明

import {Demo01Component} from './demo01/demo01.component'

在 @ngModule装饰器 的declarations数组中，添加上Demo01Component

②使用组件

就像使用html标签一样去使用（指定的选择器的名字）

练习：实现一个组件，demo02文件夹中，命名为demo02.component.ts,在该组件中，显示一个有5个元素的列表。

Day2

一、软件领域中的设计原则和设计模式

1、反面案例

不够灵活、不够健壮、反复的重复。。。

2、设计原则

设计原则：前辈帮总结好的编程技巧、经验

①YAGNI：

you aren't gonna need it

不写不需要的代码，不要把我们认为会出现的需求写到项目中来

②KISS：

keep it simple and stupid

代码越简单越好，傻瓜式，遵循命名规则，加上合理的注释

③DRY：

don't repeat yourself

不要重复自己，善于封装和重用

④高内聚低耦合：

耦合 藕断丝连

低耦合指的就是不同的模块之间关系 要尽可能的少；将降低耦合度的过程，叫做解耦

内聚：一个模块内部之间组件或者方法等之间的关系

耦合度过高会带来什么问题？

比如淘宝，有配送模块，有用户模块，有商品模块，有订单模块，如果配送模块和商品模块有着非常紧密的关系，万一配送模块出了问题，在配送模块进行升级和维护时候，一不小心改到了商品模块，就有可能导致其他问题的出现

⑤SRP

single responsibility principle

单一责任原则

指的是可以把每一个功能都封装在一个方法或者一个文件中，要尽量做到每一个文件、每一个方法都只做一件事

⑥OCP

open close principle

开闭原则

开放：扩展

关闭：修改

⑦最少知识法则

封装的功能单位要小，尽量只与自己相关（和高内聚低耦合、SRP有点类似）

昨天学习了angular的概述、组件创建和使用

二、angular所支持的常见指令

知识点回顾：

vue

v-for v-if v-else v-else-if

v-bind(简写:src) :class="{myRed:true}"

v-on( 简写 @click)

v-show

v-model 双向数据绑定的指令

在angular中，指令有很多类型，其中一种称之为结构型指令，它在使用的时候，将影响到html的结构。

1、循环指令

<any \*ngFor="let tmp of list"></any>

<any

\*ngFor="let tmp of list;let myIndex=index"></any>

作用：将遍历集合（数组、对象）中的每一个元素，

根据元素的个数，来创建多个标签（调用该指令的标签）

2、选择指令

<any \*ngIf="expression"></any>

根据表达式执行的结果的真假 来决定是否要将该元素挂载到DOM树

3、多重选择指令

\*ngSwitchCase 是多重判断的一个分支

\*ngSwitchDefault 如果不是switchCase当中任何一个，是多重判断的最后一个处理

answer = 'a';

<div [ngSwitch]="answer">

<p \*ngSwitchCase="'a'"> A </p>

<p \*ngSwitchCase="'b'"> B </p>

<p \*ngSwitchDefault> 请输入正确的答案 </p>

</div>

练习：

创建一个组件，demo04.component.ts

在这个组件中，有一个变量uName，希望在视图中根据uName来实现一个列表项（如果uName是lucy，显示'Hello lucy';如果uName是mike，显示‘ 你好，mike’；否则显示‘hello’）;

有一个学生数组（对象数组有5条数据），每一个对象都包含stuName、stuScore、stuAddr；将这5条数据显示在table中，而且只显示成绩在80分以上的数据

注意事项：

不允许把结构型指令(\*开头的指令) 用在同一个元素上边.

解决方案：是借助于ng-container来实现，ng-container在使用的过程中 并不会引入一个新的html层级

4、事件绑定的方式（圆括号）

<any (eventName)="处理函数"></any>

比如：

<button (click)="handleClick">clickMe</button>

5、属性绑定的方式 （中括号）

<any [attr]="变量"></any>

比如：

<img [src]="imgUrl"/>

<a [href]="linkUrl"></a>

动态绑定样式类或者样式

ngClass

ngStyle

<!--借助于ngStyle来实现样式的动态修改-->

<p [ngStyle]="{color: myColor}">hello,i'm a paragraph</p>

<!--借助于ngClass来实现样式类的动态绑定-->

<h2 [ngClass]="{myBlue:false}">myColor will change</h2>

扩展：如果要给一个组件指定某一个css文件，怎么指定？？

@Component({

//指定要加载的模板的页面地址

templateUrl:''

styleUrls:['./test.css','./hello.css']

})

6、插值表达式

<any>{{表达式}}</any>

双花括号、插值表达式、mustache、interpolation

---》

指的都是双花括号

将表达式执行的结果 输出到当前元素的innerHTML中

练习：

创建一个组件demo06.component.ts

该组件要求它的模板内容 存储在一个html文件中；

模板内容：有一个button，有一个p标签

功能：点击button，启动一个定时器，定时器处理函数中修改一个变量，这个变量会动态修改p标签的透明度（0~1，继续从0~1）

(click)

[ngStyle]

7、双向数据绑定的指令

ngModel指令 属于FormsModule（表单模块）；

在app.module.ts默认只依赖browserModule，如果要使用其他模块，直接import引入。

①在app.module.ts文件中 引入

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

②在app.module.ts文件中 通过imports数组来指定当前模块所依赖的其它模块

@NgModule({

imports:[FormsModule]

})

③使用双向数据绑定指令

<input type='text' [(ngModel)]="myMsg"/>

注意：

如何给表单元素监听用户的操作结果？

绑定ngModelChange事件

练习：创建一个组件demo08.component.ts

组件的模板中：

功能1：双向数据绑定

有一个复选框，有一个按钮，当复选框选中，按钮可以被点击，否则禁用

功能2：ngModelChange

有一个输入框，在输入框有一个无序列表，要求无序列表显示用户输入过的所有的信息

补充：

typescript 常见基本数据类型：

boolean/number/string/number[]/string[]/any..

定义一个整型的数据count

count:number = 0;

编写具有typescript语法的代码，浏览器默认能识别吗？

tsc :typescript compiler 将typescript转换成浏览器能识别的js的语法

总结：

①设计原则

7大设计原则

②angular提供的常见指令

\*ngFor

\*ngIf

\*ngSwitchCase

\*ngSwitchDefault

() 事件绑定

[] 属性绑定

{{}} 将双花括号中表达式的结果输出到当前元素的innerHTML

[ngClass] 动态样式类的绑定

[ngStyle] 动态样式的绑定

[(ngModel)]="kw" 前提：在app.module.ts中指定依赖于FormsModule

(ngModelChange)=""

Day3

重点：学习路由模块

一、知识点回顾：

Vue路由模块：建立url和代码片段之间的映射关系

①SPA工作原理

什么SPA？

单一页面应用程序，只有一个完整的html页面，其余都是代码片段

SPA的工作原理？

http://localhost/index.html/#/start

①解析完整的index.html,加载（一般情况，该页面都会有一个用来盛放代码片段的容器）

②解析路由地址 start

③会从路由词典中 找到start所对应的配置信息

④加载对应的代码片段 显示出来

②vue去实现SPA基本步骤

①引入vue-router.js

②指定容器 <router-view></router-view>

③创建业务所需要用到的各个组件

④配置路由词典

new Vue({

router:new VueRouter({

routes:[

{path:'/start',component:StartComponent},

{path:'/login',component:LoginComponent}

]

})

})

⑤测试

在地址栏中 取输入对应的路由地址 ，检测是否能够加载所设置的组件

③跳转的方式

①修改地址栏

②js

this.$router.push('/login')

③超链接

<router-link to="/login">去登录</router-link>

④传参的方式

①明确发送 接收方

②配置接收方的路由地址

{path:'/start',component:StartComponent}

--》

{path:'/start/:id',component:StartComponent},

③接收参数

this.$routes.params.id

④发送参数

this.$router.push('/start/10')

<router-link to="/start/20">去登录</router-link>

⑤路由嵌套

什么是路由嵌套？

在一个模板中 嵌套另一个模板

实现的方式：

①指定一个容器

<router-view></router-view>

②配置路由地址

const myRoutes = [

{path:'/login',component:LoginComponent},

{path:'/mail',component:MailComponent,

children:[

{path:'/inbox',component:InboxComponent}

]

}

]

二、angular中路由模块的使用

路由模块：建立起url和组件之间的关系

补充：在angular中，框架会自动解决模块之间依赖关系

1、实现SPA的基本步骤

①指定模块 依赖于 路由模块

在app目录下，创建一个app.router.ts,

通过插件中的快捷方式a-module-routing ，创建一个自定义模块，已经指定依赖于Routes,RouterModule

①引入组件

②在对象数组中配置

③forChild --> forRoot

④修改了导出的自定义模块类名称 AppRoutingModule

找到app.module.ts,引入AppRoutingModule并在imports数组中指定依赖于AppRoutingModule

②需要有一个容器用来盛放加载的组件

<router-outlet></router-outlet>

③创建所需要用到的组件

④配置路由

const routes:Routes = [

{path:'',component:Demo09Component}

]

⑤测试

练习：

实现一个SPA，创建一个文件夹 demo10\_spa,有两个组件

demo10\_login.component.ts->Demo10LoginComponent

demo10\_main.component.ts --》Demo10MainComponent

要求：

当路由地址为空或者login显示 Demo10LoginCompoent

当路由地址为main,显示Demo10MainComponent

当地址栏中路由地址 不是配置过的任何一个，就通过\*\*来设置要进行异常处理时 要加载的组件，

注意：一般放在数组的最后一个元素的位置

2、跳转的方式

方式1：直接修改地址栏（开发人员自己测试）

方式2：通过js的方式

①import {Router} from '@angular/router'

②在构造函数中 完成变量定义

constructor(private router:Router){

}

③调用方法来实现跳转

jump(){

this.router.navigateByUrl('/login');

}

方式3：通过超链接

<a routerLink="/login">跳转到login</a>

补充：

在angular如何来通过路由模块，实现返回上一页的功能？

①引入Location

import {Location} from '@angular/common'

②定义Location类型的变量

constructor(private myLocation:Location){

}

③实现返回上一页

this.myLocation.back();

通过Router中的navigate方法来跳转？

①引入Router

import {Router} from '@angular/router'

②定义Router类型的变量

constructor(private router:Router){

}

③跳转

this.router.navigate(['/login'])

练习：实现一个SPA

创建一个文件夹demo12\_jump，在该文件夹中创建3个组件，

notFound.component.ts

(有一个h1:404，有一个button：返回上一页)

-->NotFoundComponent

demo12List.component.ts

(列表页面：渲染一个有5个元素的列表，要求点击列表项 通过js的方式跳转到详情页面；超链接，点击也可以跳转到详情页面)

-->ListComponent --》myList

demo12Detail.component.ts（

详情页面：显示h1标签；超链接，故意给个错的path）

-->DetailComponent -->myDetail

3、传参

①理清楚发送 接收

Demo13StartComponent Demo13DesComponent

②配置接收方

找到接收方 所对应的路由地址

{path:'myDes',component:Demo13DesComponent}

-->

{path:'myDes/:uname',component:Demo13DesComponent}

③接收参数

import {ActivatedRoute} from '@angular/router'

constructor(private aRouter:ActivateRoute){

}

this.aRouter.params.subscribe((data)=>{

this.rcvMsg = data.uname;

})

④发送参数

this.myRouter.navigateByUrl('/myDes/123')

<a [routerLink]="'/myDes/'+myInput">跳转</a>

补充：

异步请求的方式？

①ajax

②事件绑定

③promise

④rxjs 是由微软所推出的基于消息的订阅机制

observable/subscribe

传参的另外一种方式：

①明确发送方和接收方

②到接收方准备接收参数

ActivatedRoute

③ 发送参数

this.router.navigate(['/myWork',{id:2,uname:'zhangsan'}])

4、路由嵌套

{

path:'mail',

component:MailComponent,

children:[

{path:'',component:InboxComponent},

{path:'inbox',component:InboxComponent},

{path:'outbox',component:OutboxComponent}

]

},

和Vue中不一样的地方：

@Component({

selector: 'mail',

template:`

<a routerLink="/mail/inbox">收件箱</a>

<a routerLink="/mail/outbox">发件箱</a>

<a routerLink="/mail/trash">垃圾箱</a>

<router-outlet></router-outlet>

`

})

综合练习：

创建一个文件夹demo15\_homework，创建3个组件

check.component.ts

CheckComponent /storeCheck

有一个按钮，点击发送10给detail

detail.component.ts

DetailComponent /storeDetail

detail接收数据显示在该页面，放上一个超链接，点击跳转到send

send.component.ts

SendComponent /storeSend

button 点击跳转到check

Day4

学习安排：

1、组件间通信

2、内置管道和自定义管道

3、自定义指令

4、自定义服务

5、网络服务

6、路由守卫

一、回顾Vue组件间通信的方式

方式1：props down

Vue.component('parent',{

data:function(){

return {

sonName:'zhangsan'

}

},

template:`

<son :myName="sonName"></son>

`

})

Vue.component('son',{

props:['myName'],

template:`

<h1>{{myName}}</h1>

`

})

方式2：events up

Vue.component('parent',{

data:function(){

return {

sonName:'zhangsan'

}

},

methods:{

recvMsg:function(data){

}

},

template:`

<son @customEvent="recvMsg"></son>

`

})

Vue.component('son',{

methods:{

sendToFather:function(){

this.$emit('customEvent','hello father');

}

},

template:`

<button @click="sendToFather">send</button>

`

})

方式3：$refs $parent

Vue.component('parent',{

template:`

<son ref="mySon"></son>

`,

mounted:function(){

//得到子组件的实例 this.$refs.mySon

}

})

Vue.component('son',{

methods:{

getFather:function(){

//得到父组件的实例 this.$parent

}

},

template:`

<button @click="getFather">send</button>

`

})

方式4：兄弟组件通信

var bus = new Vue();

Vue.component('xiong-da',{

mounted:function(){

bus.$on('msgToXiongDa',function(data){})

}

})

Vue.component('xiong-er',{

methods:{

sendToXiongDa:function(){

bus.$emit('msgToXiongDa','...')

}

}

})

二、组件间通信

1、props down

在父组件中：

<son myName="zhangsan"></son>

在子组件中：

import {Input} from '@angular/core'

export class Demo16SonComponent{

@Input() myName="";

}

2、events up

event:

①绑定事件

在父组件中：

步骤1：定义一个事件处理函数

recvMsg(data:any){

}

步骤2：绑定事件处理函数

<son (customEvent)="recvMsg($event)"></son>

②触发事件

在子组件中：

步骤1：

import {Output,EventEmitter} from '@anular/core'

步骤2：

在该组件类中

@Output() customEvent = new EventEmitter();

步骤3：触发事件

this.customEvent.emit('hello father');

练习：

创建一个文件夹 demo17\_tongxin

demo17-main.component.ts

Demo17MainComponent 路由地址：demo17Main

demo17-header.component.ts

Demo17HeaderComponent

功能要求：

在Demo17HeaderComponent组件中，定义一个判断用户是否登录的方法，返回true、false

在Demo17MainComponent组件中 是由Demo17HeaderComponent和一个列表构成的，如果用户已经登录，加载列表（模拟3条数据），否则显示一个<h1>提示请登录

核心步骤：将子组件中 是否已登录 传递给父组件

触发事件可以在ngOnInit中触发

3、父组件与子组件通过本地变量互动

<p>{{mySon.name}}</p>

<son #mySon></son>

三、angular管道

1、内置的管道

date/json/uppercase/lowercase/

number/currency/percent/slice

number:[整数部分保留最小位数].[小数部分保留最小位数]

比如：

{{price | number:'3.4'}}

currency:currencyCode:symbolDisplay:小数点

比如：

{{price | currency:'USD':true:'3.4'}}

percent：小数点

比如：

{{value | percent}}

{{value | percent:'3.4'}}

2、自定义管道的实现

在angular这个框架中 有8个核心概念。

模块、组件、模板、元数据、指令、服务、依赖注入、数据绑定。

步骤1：创建一个文件 demo18.pipe.ts

步骤2：在demo18.pipe.ts文件：

import {Pipe,PipeTransform} from '@angular/core'

@Pipe({

name:'myCurrency'

})

export class Demo18Pipe implements PipeTransform{

transform(value: any, ...args: any[]): any {

//必须有返回值

//value管道操作符前面传来的数据

}

}

步骤3：在模块中 声明管道

import {Demo18Pipe} from '...'

@NgModel({

delcarations : [Demo18Pipe]

})

步骤4：调用管道

{{ expression | myCurrency}}

四、angular服务

组件：一个具有特定功能的，可被反复使用的视图

管道：本质是一个方法，用来实现对于数据的筛选 过滤 格式化

服务：把具有公共都需要用到的方法 封装成一个服务

术语：

依赖注入:将我们需要用到的东西 注入 到当前的对象中去用，是通过提供商的设置来实现的

提供商：providers，将依赖注入到当前的对象中

providers:[LogService,ShowService,HttpService]

单例模式：在angular中 服务就是单例模式的类；单例模式指的在一个应用程序中，这个对象只保留单个实例。

7大设计原则：

SRP/YANGNI/OCP/kISS/高内聚低耦合/DRY/LKP

23种设计模式：

总体来说设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

常见：单例模式、观察者模式、工厂模式、抽象工厂模式。。。

1、服务的创建

步骤1：创建一个log.service.ts的文件

步骤2：引入Injectable

import { Injectable } from '@angular/core';

步骤3：在服务类中 定义需要用到的方法和数据

@Injectable()

export class LogService {

isDev:boolean = false;

constructor() { }

print(msg:string){

if(this.isDev)

{

console.log(msg);

}

}

}

2、服务的使用

在组件中调用服务的方式：

①在组件中 引入 对应的服务类

import {LogService} from './log.service'

②在组件的元数据添加一个Providers（用来指定需要提供哪些服务）

@Component({

providers:[LogService]

})

③得到服务的实例

constructor(private myService:LogService){

}

④调用服务的方法

this.myService.\*\*\*\*();

练习：创建一个自定义的服务

HeartBeatService,实现两个方法，一个方法开始心跳：启动了周期性定时器，每隔0.5s，在控制台输出一句话“正在心跳”

一个结束心跳：结束周期性定时器

创建一个文件夹demo20，创建一个组件 demo20.component.ts，导出的类 Demo20Component,路由地址 demo20

在这个组件中有两个按钮，希望能够分别调用服务中的开始和结束心跳的方法。

Day5

学习安排：

1、常用的网络服务的封装方法

2、路由守卫

3、项目整体架构

4、动手开始做项目

一、封装一个网络服务

DI： Dependency Injection 依赖注入

ngModel FormModule

router RouterModule

http HttpModule

创建一个网络服务并去调用完整的过程？

①创建一个网络服务

//a-service-http

②在服务中封装实现网络请求的方法

在服务类中定义一个方法

sendRequest(myUrl:string){

//发起网络请求

return this.http.get(myUrl)

.map((response: Response) => response.json());

}

③在组件中准备调用

import {MyHttpService} from './myhttp.service'

在组件的元数据中指定提供商

@Component({

providers:[MyHttpService]

})

在构造函数中得到该服务的实例

constructor(private httpService:MyHttpService){}

④使用服务

this.httpService

.sendReuqest("./data.json")

.subscribe((data)=>{

})

总结异步请求方式：

ajax、callback（回调函数）、promise、RxJS

二、路由守卫

路由守卫：保护指定的路由，当需要切换到该路由地址的时候，会根据指定的值的true、false来决定是否可以跳转。

服务知识的补充：

服务可以注入到一个组件，也可以注入到一个模块中。

如果服务通过不同组件的providers，注入到不同的组件，那么服务是被创建了很多次，每一个都是一个单独的实例，都没有关系

如果服务注入到了一个组件中，那么这个组件 由有其它的子组件构成，那么在子组件中如果引入服务类（不会通过providers去指定），那么所实例化的服务类对象 是同一个实例对象。

如果服务是注入到了一个模块中，那么该模块任何一个组件所import和实例化得到的服务都是同一个实例

路由守卫：

步骤1：创建一个服务can-enter.service.ts

//a-guard-can-activate 快速创建一个路由守卫的服务

步骤2：在这个服务canActivate方法中，处理是否能够访问,通过返回值是true（允许访问）/false(禁止访问)

步骤3：在app.module.ts对应的模块中 指定一个提供商属性

import {CanEnterGuard} from ''

@NgModule({

providers:[CanEnterGuard]

})

步骤4：

在app.router.ts对应的路由配置对象中，指定的属性

{

path:'admin',

component:AdminComponent,

canActivate:[CanEnterGuard]

}

三、自定义指令

在vue中要想实现一个自定义指令

new Vue({

directives:{

change:{

bind:function(el){},

update:function(el){}

unbind:function(){}

}

}

})

在angular中实现一个自定义指令的步骤：

①创建一个指令类

//a-directive

import {Directive} from '@angular/core';

@Directive({

selector:[test]

})

export class TestDirective{}

②到模块中声明

import {TestDirective} from ''

@NgModule({

declarations:[TestDirective]

})

③使用

<h1 test></h1>

练习：考察自定义指令

创建一个自定义指令 changBgColor

<h3 changeBgColor="#ff0000">

hello custom directive

</h3>

指令接收到数据，修改背景色传递来的数据。

Day6

我们是这样写 Angular 应用的：

用 Angular 扩展语法编写 HTML 模板（angular指令、管道、插值表达式等），

用组件类管理这些模板（通过组件来指定模板显示），

用服务添加应用逻辑（将应用中 需要经常用到的逻辑封装在服务中，方便服用），

用模块打包发布组件与服务（组件、服务需要在一个模块中去创建去声明）。