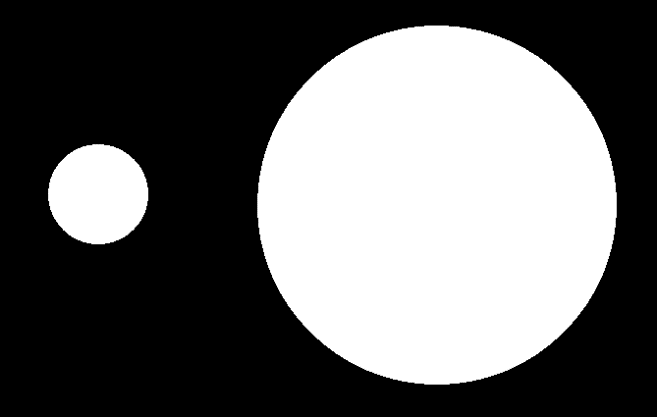
# day01



CSS => Less/SCSS

JS => ES => TS(TypeScript)

DOM => jQuery => MVVM

CSS => SCSS(编译器)

JS => TS(编译器)，是JS的超集

**如何自学 = 基础知识 + 官网手册(零散时间)**

如何自学一门新语言？

如何自学一门新框架？

<https://cloud.tencent.com/developer/devdocs>

## 1.设计原则

(1)YAGNI

You Aren't Gonna Need It，你不会需要它；不写不需要的功能

(2)KISS

Keep It Simple and Stupid，让你的代码越简单/傻瓜越好

(3)OCP

Open Close Principle，开闭原则，对外界修改封闭(不允许修改已有代码)，对外界的扩展开放

(4)High Cohesion，Low Coupling

高聚合，低耦合；功能相关代码紧密在一起；功能不相干代码拆分越明确越好

(5)迪米特法则/最少知识原则

一个对象/组件，数据/操作越少越好

## 2.Angular

Angular，由Google在2009年创建的MVVM框架，适用于中大型的企业级SPA应用。

V1.x官网：https://angularjs.org/ Angular.js用JS编写

V2.x~8.x官网：https://angular.io/ Angular用TS编写

V2.x~8.x中文网：https://angular.cn/

提示：V1到V2的升级变化非常大！ V1支持SCRIPT直接引入和脚手架方式；V2开始只支持脚手架方式。

注意：Angular 8.x要求Node.js版本必须是10.9以上！

|  |
| --- |
| 提示：如果使用NPM从官方网站下载NPM包总是失败，可以把默认下载地址改为国内淘宝网镜像：  查看当前下载地址： npm config get registry  默认值为： https://registry.npmjs.org/  修改为淘宝网NPM镜像： npm config set registry=https://registry.npm.taobao.org/ |

## 3.创建第一个Angular项目

①安装Node.js和NPM

Node.js版本必须>=10.9

②安装全局的Angular脚手架工具

npm i -g @angular/cli

默认会在C:\Users\web\AppData\Roaming\npm目录下安装ng.cmd可执行文件及其相关文件

③运行全局脚手架，创建一个Angular空白项目

ng new myngapp01

此步需要联网从NPM官网下载第三方模块，如果总是失败可以拷贝他人下载好的空白项目

④进入项目根目录，运行该项目（Node.js项目）

npm start 或者 ng serve

|  |
| --- |
| Vue.js项目的主配置文件： vue.config.js  Angular项目的主配置文件： angular.json |
| ES6中有“模块”的概念： export/export default、 import..from..  Vue.js中无自己的“模块”概念——项目是由自定义组件构成；  Angular中有自己的“模块”概念——项目是由自定义组件构成，每个组件都要放在一个特定的Module中 |

## 4.Angular项目引导启动流程

①根据angular.json查找系统运行入口文件：main.ts

②main.ts启动主模块： AppModule

③在主模块中引导启动主组件：AppComponent

④主组件中声明了模板和样式，最终渲染在index.html中 <app-root/>

## 5.Angular核心概念之一 —— 组件

Component：组件，就是一段可以复用的页面片段，如轮播广告、手风琴、下拉菜单.......；每个组件内有自己专用的HTML片段、CSS样式、JS数据和操作。

Component = Template + Style + Script

提示：Angular中每个组件(Component)必须处于某个模块(NgModule)中！！

自定义组件的步骤：

①创建一个ts文件

例如： src/app/myc01.ts

②ts文件中声明并导出一个class

@Component({

selector: 'app-myc01', //标签名

template: '<h1>我的组件01</h1>' //组件模板

})

export class MyComponent01 { }

③在某个模块中声明该组件，否则无法使用该组件

@NgModule({ declarations: [ MyComponent01 ] })

④在当前模块内的其它组件的模板中使用该组件

//app.component.html

<app-myc01></app-myc01>

练习：创建一个自定义组件MyComponent02，其对应的元素为app-myc02；模板内容随意；试着在前面两个组件中使用这第三个组件

练习：创建一个自定义组件MyComponent03，其对应的元素为app-myc03；模板内容随意；试着在前面三个组件中使用这第四个组件

练习：创建一个文件夹myc05，其中自定义组件MyComponent05，其对应的元素为app-myc05；模板/样式独立出来都放在此文件夹下；试着在主组件中使用这个组件

**提示：可以使用Angular脚手架提供的 ng g component 组件名 来自动的创建一个组件所需要的文件；若没有成功安装全局命令ng，也可以使用 npx ng g component 组件名 。 （NPX是Node.js安装包提供的一个命令，可用于执行node\_modules下的可执行文件）**

## 6.Angular核心概念之二 —— 数据绑定

Data Bingding：把脚本中的数据(称为Model模型)，绑定到HTML(称为View视图)，未来只要Model发生改变，则View自动更新。

Angular中数据绑定的呈现方式：

①插值语法/大胡子语法，绑定到innerHTML：{{ }}

<any>{{expression}}</any>

②属性绑定：[ ]

<any [attr]="expression">

<any attr="{{expression}}">

③事件绑定：( )

<any (event)="fn( )">

注意：事件绑定中函数后的( )不能省略！

说明：(1)插值语法中可以使用算术运算吗？比较运算符？逻辑运算符？三目运算符？调用对象的方法吗？自增/自运算符？创建新对象吗——{{}}中不能出现new？

练习：使用数据绑定实现如下组件：myc11，购买数量增减



数据绑定相关的指令：

①循环绑定：\*ngFor

<li \*ngFor="let tmp of scoreList; let i=index;"> {{i}} - {{tmp}} </li>

②选择绑定：\*ngIf

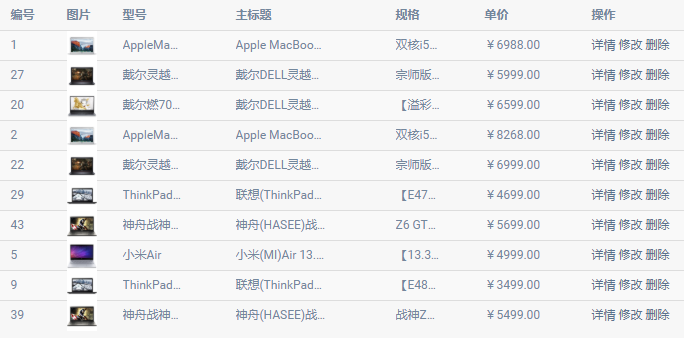
③样式绑定：[ngStyle]、[ngClass]

## 课后任务：

(1)比较Vue.js和Angular的不同

|  |  |  |
| --- | --- | --- |
|  | Vue.js | Angular |
| 插值表达式 |  |  |
| 属性绑定 |  |  |
| 事件绑定 |  |  |
| 循环指令 |  |  |
| 选择指令 |  |  |
| 样式指令 |  |  |

(2)创建组件，包含Model数据：productList: [ (),(),()...() ]，绑定到如下的表格中，要求：点击“删除”可以实现模型数据的删除功能



# day02

## 复习：

(1)环境搭建

①升级Node.js到10.9+ node/npm/npx

②全局下载Angular脚手架工具

npm i -g @angular/cli

③运行脚手架创建空白项目

ng new myngapp02

④进入空白项目，运行当前项目：package.json/scripts

cd myngapp02

npm start

⑤使用客户端浏览器访问

http://127.0.0.1:4200

(2)Angular核心概念之一——组件

可以复用的页面片段，

Component = Script + Template + Style

Angular中的组件是多文件组件： .ts + .html + .css/.scss

注意：组件必须在某个模块(AppModule)中声明！

使用快捷命令创建： ng g component 组件名

(3)Angular核心概念之二——数据绑定

①innerHTML绑定：{{ }}

②属性绑定：[ attr ] = ""

③事件绑定：( click ) = "fn( )"

④循环绑定指令：\*ngFor

⑤选择绑定指令：\*ngIf

⑥选择绑定指令：[ngSwitch] \*ngSwitchCase \*ngSwitchDefault

⑦样式绑定指令：[ngStyle]

⑧样式绑定指令：[ngClass]

⑨双向数据绑定指令：

|  |
| --- |
| 面试题：Angular中指令分为哪几类：  **(1)组件指令**  <myc01></myc01>  Component extends Directive，组件继承自指令，组件是有模板的指令  **(2)结构型指令**  可以影响当前的DOM结构的指令；所有结构型指令都以 \* 开头  **(3)属性型指令**  不会影响DOM结构，只会影响当前元素的特征，如样式；所有的属性型指令都用 [ ] 括起来！ |

## 1.Angular核心概念之三——指令(Directive)

指令：是一种特殊的模板页面内容，可以对页面执行特殊的处理；

例如： <any \*ngFor="">

Angular中常见指令：

①循环绑定指令： \*ngFor

<any \*ngFor="let tmp of 集合对象">

②选择绑定指令： \*ngIf

<any \*ngIf="expression">

提示：\*ngIf会影响DOM结构

③选择绑定指令： [ngSwitch] \*ngSwitchCase \*ngSwitchDefault

<any [ngSwitch]="变量名">

<any \*ngSwitchCase="值">....</any>

<any \*ngSwitchCase="值">....</any>

...

<any \*ngSwitchDefault>....</any>

</any>

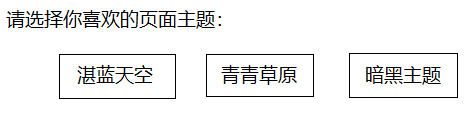
④样式绑定指令：[ngStyle]

<any [ngStyle]="obj">

⑤样式绑定指令：[ngClass]

<any [ngClass]="obj">

练习：创建新的组件myc02-directive，实现某块区域样式切换效果：



## 2.(了解)如何自定义指令

提示：可使用工具 ng g directive 指令名 快速的创建一个指令

<div appNeedStrong>....</div>

@Directive({ selector: 'appNeedStrong' })

export class MyDirective {

contructor( el: ElementRef){

el.nativeElement.xxx....

}

}

## 3.Angular中的双向数据绑定

**方向1：Model => View**

**方向2：View => Model**

Angular中实现双向数据绑定的方法：

<input [(ngModel)]="userName">

**提示：ngModel指令处于FormsModule，必须在当前模块中声明导入才能使用：**

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

@NgModule({

imports: [ FormsModule ]

})

如果想监视模型数据的改变（就像Vue.js中的watch函数），可以使用 ngModelChange事件：

<input [(ngModel)]="userName" (ngModelChange)="fn( )">

练习：创建密码输入框，随着用户的输入，后面的提示信息不停的在改变：

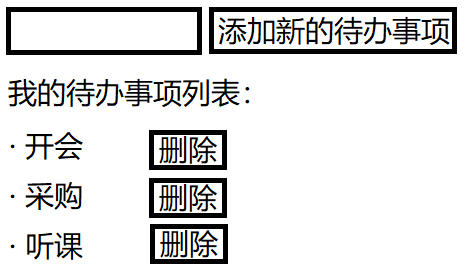
"密码不能为空"

"密码长度不能少于6位"

"密码长度合法"

"密码长度不能超过12位"

练习：使用Angular实现 TodoList



## 4.Angular核心概念之四——管道

Vue.js中的过滤器(filter)： <p>{{ 1 | sex('zh') }}</p>

Angular中的类似的概念称为管道(pipe)：<p>{{ 1 | sex:'zh' }}</p>

Vue.js没有内置任何过滤器；但Angular内置了很多好用的管道：

①lowercase：把数据转换为小写形式

②uppercase：把数据转换为大写形式

③slice：获取字符串或数组中的一部分

④json：把对象转换为JSON字符串

⑤date：把日期/数字转换为特定格式的日期字符串

⑥number：数字格式化(每三位加逗号，并指定小数位数)

⑦currency：把数字以货币形式显示

## 课后任务：

(1)对比Vue.js和Angular中数据绑定方法

(2)对比Vue.js和Angular中所有的指令

(2)创建组件 xz-register，实现学子商城注册页面中所有的输入校验和注册提交功能（需要借助XHR访问后台数据API）

# day03

|  |
| --- |
| TypeScript语法知识： |
| (1)类型声明：  class中的成员（属性、方法）、参数的参数都可以声明类型：  class Book{  price: number = 12.5  add(n: number): void{ } } |
| (2)TS中的新类型：接口(interface)  接口是一种特殊的class，用于规定一个对象“**必须具备**”哪些功能  interface Runnable{  run( );  stop( );  }  class Car implements Runnable{  run( ){ .... }  stop( ){ .... }  } |

## 复习：见思维导图

## 1.自定义管道：

工具命令： ng g pipe 管道名

@Pipe({

name: 'sex'

})

class SexPipe {

transform(val, args){

return ...;

}

}

练习：创建一个管道：zzmm，可以把数字1/2/3分别转换为“党员/中国共产党党员”、“团员/中国共青团团员”和“群众/中国人民群众”

## 2.(重点/难点)父子组件间的数据传递

ng g component parent

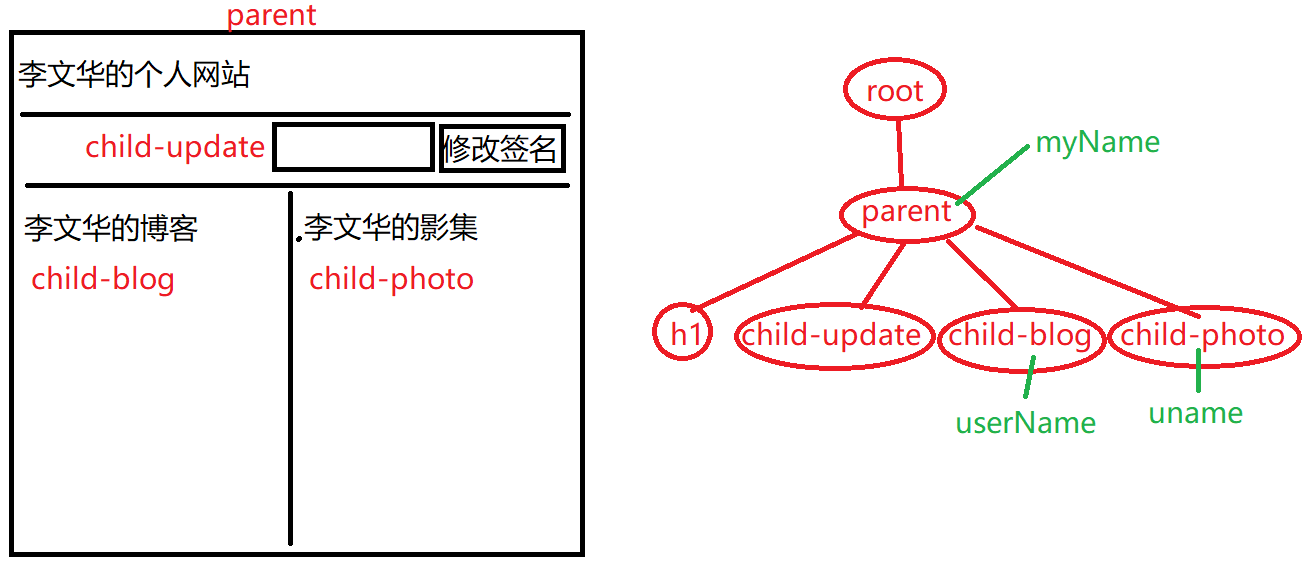
ng g component child

--------------------------------------------------

parent.component.html:

<app-child></app-child>

--------------------------------------------------



**(1)父组件给子组件传递数据： 父=>子 —— Props Down**

(1.1)子组件声明自己专有的属性

@Input() //Input装饰器把下面的属性变为“输入型属性”

userName: string;

(1.2)父组件使用子组件的专有属性赋值——值为父组件的模型数据

<app-child [userName]="myName">

**(2)子组件给父组件传递数据： 子=>父 —— Events Up**

(2.1)子组件声明并触发事件，触发时携带自己的数据

@Output() //声明输出型属性

unameEvent = new EventEmitter();

.....

this.unameEvent.emit( 123 );

(2.2)父组件监听子组件的事件，并提供处理函数接收事件数据

<app-child (unameEvent)="doEvent(**$event**)" >

...

doEvent( data ){ .... }

## 3.服务和依赖注入

Angular认为：

**Component：**组件，仅应该只负责视图，只参与展示；与展示无关的语句都应该剥离出去；例如：做日志、计时、网络数据访问...

**Service：**服务，就是一个简单的对象，负责执行从组件中剥离的与视图无关的操作，例如：做日志、计时、网络访问....

创建服务： ng generate service 服务名

|  |
| --- |
| inject：打针，注入  injectable：能被注入给别人的 |
| Angular创建对象的两种方式：  **(1)手工创建**  var logger = new LoggerService();  logger.log('用户增加了一个商品');  **(2)依赖注入(Dependency Injection)**  constructor(**logger:LoggerService**){  logger.log('用户增加了一个商品')  }  在构造方法中声明需要依赖某个对象，且该对象是“可以被注入的(@Injectable)”,那么Angular就会自动创建依赖的对象，并注入给当前构造方法 |

今日练习：使用Angular提供的异步请求服务，完成学子商城“商品列表”

(1)API：http://www.codeboy.com/data/product/list.php?pno=2&kw=

(2)Angular提供了一个网络访问**服务**： HttpClient

1)在主模块中引入HTTPClientModule模块

import { HttpClientModule } from '@angular/common/http'

....

imports: [ HttpClientModule, ],

2)在组件中注入依赖的HTTPClient服务

import { HttpClient } from '@angular/common/http';

class XxxComponent{

http = null;

constructor( http: HttpClient ){

this.http = http;

}

//加载商品

loadProduct(){ //3)在组件方法中使用被注入的httpClient服务

~~this.http.get(url).then( (res)=>{ } )~~

this.http.get(url).subscribe( (res)=>{ } )

}

}

# day04

|  |
| --- |
| TypeScript知识点——强类型语言： |
| 1.类型声明：  class Book{  price: number = 1.5  author: any = 'Tom'  work( tool: number[] ): void{  }  } |
| 2.接口：是一种特殊的class，用于规定一个class必须提供特定的方法  interface Runnable{  run()  stop()  }  class Car implements Runnable{  run(){ }  stop(){ }  } |
| 3.类成员(属性/方法)的访问修饰符  ① private：私有成员，只能在当前类内部使用；class中的属性一般都应该声明为私有的或被保护的！  ② protected：被保护的，只能在当前类内部以及子类中使用  ③ public：默认值！公共成员，可以被任何类使用；class中的方法一般声明为公共的！ |
| 4.类属性的声明/赋值方式  class Book{  private price: number = 0  public constructor( price: number ){  this.price = price  }  }  ----------------------------------------------------  上面的代码可以简写为：  class Book{  //构造方法中的形参前加private/protected/public此形参自动变为一个同名属性，且自动赋值  **public constructor( private price: number ){**  //此处无需写 this.price = price  **}**  }  var book1 = new Book( 100 ) |

## 复习：

**父组件给子组件传递数据： Props Down**

parent.component.html:

<h1>这里是父组件</h1>

<app-child [uname]="parentName"></app-child>

parent.component.ts:

parentName: '父亲的名称'

------------------------------------------------

child.component.ts:

@Input( )

private uname: string

**子组件给父组件传递数据： Events Up**

parent.component.html:

<h1>这里是父组件</h1>

<app-child (childEvent)="doChildEvent($event)"></app-child>

parent.component.ts:

doChildEvent( data ){ ... }

------------------------------------------------

child.component.ts:

private childName: string = '孩子的名称'

@Ouput()

private childEvent = new EventEmitter()

doClick(){

this.childEvent.emit( this.childName )

}

## 1.核心概念之六 —— 服务和DI

Component：负责视图的数据绑定/事件处理

Service：负责从组件中剥离的与视图无关的任务，如日志、计时、网络访问....

**注意：Angular中的依赖注入是基于构造方法的参数类型，而与先后顺序无关**

创建服务：

@Injectable( {

//可以被注入给某个组件

//Injector/Provider：注入器/服务提供者，负责创建服务对象并注入给组件，Angular会自动为每个服务创建必需的注入器

} )

export class LoggerService{

}

|  |
| --- |
| 面试题：Angular中的Service可以在哪里被提供/Service的提供者/注入器有哪些？  **方式1：声明服务时提供——根模块中提供的服务对象是整个应用中“单例的”**  @Injectable({  providedIn: 'root'  })  export class LoggerService{ }  **方式2：在模块中提供——在当前模块中的所有组件共用同一个服务对象**  @Injectable()  @NgModule({  providers: [ LoggerService ]  })  **方式3：在组件声明中提供——该服务仅能作用于当前组件，且每个组件都有自己专有的服务对象**  @Component({  selector: 'app-myc01',  template: ' ',  providers: [ LoggerService ]  }) |

|  |
| --- |
| 面试题：前端技术中有哪些异步请求方案？  (1)原生XHR： 1234  (2)jQuery封装： $.ajax( ) 基于回调函数=>回调地狱  (3)Axios：底层还是XHR，基于Promise，可以避免回调地狱  (4)Angular HttpClient服务：底层还是XHR，基于Observable  (5)ES2016新方案：Fetch：底层不是XHR！就是fetch对象！不支持请求打断、请求排队！ |

## 2.Angular中异步请求服务器端数据

HTTPClient Service：是Angular官方提供的异步请求工具

使用步骤：

(1)在主模块中引入HttpClientModule——会提供HttpClient服务的注入器

@NgModule({

imports: [ HttpClientModule ]

})

(2)在需要使用异步请求模块中声明依赖HttpClient服务

constructor(private http: HttpClient){}

(3)调用HttpClient提供的异步数据请求服务

this.http.get( )

this.http.post( )

this.http.put( )

this.http.delete( )

马走日象走田 => 双炮枪等招数

JS语言基础语法 => JS设计模式(23+1)

## 3. Rx.js和Observable对象

Angular中HttpClient服务底层基于Rx.js第三方模块

官方：RxJS 是 Reactive Extensions for JavaScript 的缩写，是一个基于可观测数据流在异步编程应用库。RxJS 是 Reactive Extensions 在 JavaScript 上的实现。一般说到RxJS，都会讲他是基于流的响应式的结合观察者和迭代器模式的一种库。

**Observer观察者模式（也称为“订阅-发布”模式）：**

乙方声明“订阅(subscribe)”甲方；

在未来的某个不确定时间，甲方“发布(publish)”新消息，乙方会立即接到通知。

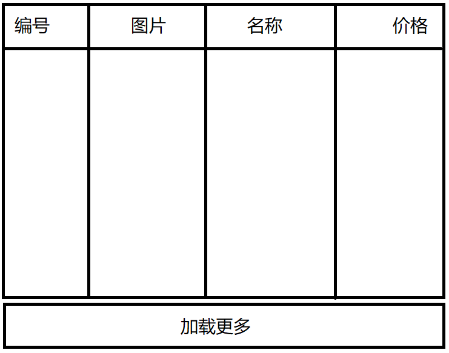
HttpClient服务采用了“观察者/订阅-发布模式”，其最核心对象为：

let obj = new Observable( ); //可被观察的对象

//可以关注/订阅“可被观察的”对象

obj.subscribe( ( )=>{ //收到订阅消息时的回调函数 } )

练习：创建组件：ProductList，点击“加载更多”则异步请求更多数据，呈现在表格中



## 4.Angular中组件的声明周期钩子函数

Hooks Function：声明好的特定的函数，到了指定的时间点，就会被自动执行

LifeCycle Hooks Function：在组件的不同生命阶段会自动执行的函数

面试题：Angular中组件的生命周期钩子函数按顺序有：

(0)constructor( )：构造方法，执行且仅执行一次

(1)ngOnChanges( )：组件的输入属性值发生赋值或改变

**(2)ngOnInit( )：组件正在初始化，一般用于组件刚加载完成时执行的业务操作，如获取页面数据；执行且仅执行一次**

(3)ngDoCheck( )：组件正在执行变化检查

(4)ngAfterContentInit( )：组件内容初始化之后

(5)ngAfterContentChecked()：组件内容被重新检查后

(6)ngAfterViewInit( )：组件视图初始化之后

(7)ngAfterViewChecked()：组件视图被重新检查后

(8)ngOnDestroy( )：组件即将从DOM树上销毁，用于释放定时器、取消订阅...执行且仅执行一次

## 5.核心概念之七 —— 路由和SPA应用

Single Page Application，单页应用——整个应用中只有一个完整的HTML页面，其它所有的“页面”其实都是一段HTML片段



SPA应用的优点：

(1)DOM树只需要创建一次，“页面切换”只是在切换部分元素

(2)便于实现“过场动画”——多页应用不可能做到

SPA应用的不足：

(1)不便于实现SEO优化

SPA应用的核心——**路由词典**(把一个地址和一个组件对应起来)：

[

{ path: 'index', component: ...}

{ path: 'product/list', component: ...}

{ path: 'user/login', component: ...}

.....

]

SPA应用的原理：框架根据客户端请求的路由地址，异步加载对应的组件内容，替换之前的组件内容。

## 6.使用Angular中的路由步骤：

(0)提前准备好路由组件

ng g component index 首页

ng g component productList 商品列表

ng g component productDetail 商品详情

(1)在根模块中创建路由词典

var routes = [ {path:'product/list', component: ...} ]

(2)在根模块引入路由模块，注册路由词典

@NgModule({

imports: [ RouterModule.forRoot(routes) ]  
 })

(3)在根组件的模板中声明路由组件的占位符

<router-outlet></router-outlet>

(4)让客户端请求路由地址

http://127.0.0.1:4200/product/list

## 课后任务：

(1)比较Vue.js和Angular中的声明周期钩子函数

(2)比较Vue.js和Angular中的路由的使用过程

(3)查手册，查找Angluar中的如下问题的实现方法：

①默认路由和404路由

②路由跳转

③路由传参

④路由嵌套

# day05

## 复习：见思维导图

## 练习：

0)创建如下四个组件：

ng g component index

ng g component productList

ng g component productDetail

ng g component userCenter

ng g component notFound

1)创建路由词典，为每个组件分配访问路径

index => index

productList => product/list

productDetail => product/detail

userCenter => user/center

let routes = [ {path:'', component: .. },... ]

2)在根模块中注册路由词典

RouterModule.forRoot( routes )

3)在根组件中声明路由出口

app.component.html: <router-outlet></router-outlet>

4)使用客户端访问每个路由地址，测试路由功能实现

http://127.0.0.1:4200/user/center

## 1.路由词典配置

let routes = [

{path:'', component:IndexComponent},

{path:'index', component:IndexComponent},

{path:'product/list', component:ProductListComponent},

{path:'product/detail', component:ProductDetailComponent},

{path:'user/center', component:UserCenterComponent},

{path:'\*\*', component:NotFoundComponent},

]

注意：路由词典中的路由path不能以 '/' 开头！但是，路由跳转时指定的路径最好都以 '/' 开头！

## 2.路由跳转

从一个路由地址跳转到另一个有两种方法：

方式1：编程方式

contructor( private router: Router ){ }

...

this.router.navigateByUrl( '...' )

方式2：模板方式

<any routerLink="...">

注意：routerLink指令可以用在任何元素上，如DIV、A、BUTTON

## 3.路由参数

在路由的path属性中，有些部分固定不变，有些部分需要动态变化：

{ path: 'product/detail/20', component: ... }

{ path: 'product/detail/23', component: ... }

{ path: 'product/detail/35', component: ... }

-----------------------------------------------------------------

{ path: 'product/detail/**:pid**', component: ... }

路由参数：路由地址中的变量

使用步骤：

①路由词典中设置 路由参数：

{ path: 'product/detail/**:pid**', }

②路由跳转时提供 路由参数值：

<any routerLink="/product/detail/35">

③在目标组件中读取当前路由的参数

constructor( private route: ActivatedRoute ){ }

ngOnInit(){

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

data.pid 就是路由参数的值

})

}

午间任务：完整的实现“商品列表”和“商品详情”两个组件的功能：

获取商品列表的服务器端API:

http://www.codeboy.com/data/product/list.php

获取商品详情的服务器端API:

http://www.codeboy.com/data/product/details.php?lid=35

## 4.路由嵌套

在一个路由组件内部，有部分内容固定，另一区域中的内容可以切换不同的子组件 —— 嵌套路由

路由词典：

let routes = [

{

path: 'user/center',

component: UserCenterComponent,

children: [

{ path: 'myinfo', component: ...},

{ path: 'headpic', component: ...},

{ path: 'security', component: ...},

]

}

]

路由出口：

app.component.html: <router-outlet></router-outlet>

user-center.component.html: ....<router-outlet></router-outlet>....

练习：为“用户中心”创建嵌套路由

(0)创建必需的子组件

ng g component myInfo

ng g component headPic

ng g component securityManagement

(1)修改路由词典，为“用户中心”路由添加嵌套路由

(2)修改“用户中心”模板，添加嵌套路由的出口

## 5.路由守卫

Guard：护卫，守护者

有些路由地址只有在特定条件满足的情况下才允许访问，不满足的情况下禁止访问——如是否登录、是否充值、是否满足时间段限制....

对访问条件是否满足而进行设置，满足的话，就让访问路由组件；否则就不让访问 —— 这种对象成为“路由守卫对象”。

使用方法：

①创建一个路由守卫：

@Injectable({ providedIn: 'root' })

class XxxGuard {

canActivate(){ //组件是否允许被激活

......

return true / false ;

}

}

②使用路由守卫

[{

path: 'user/center',

component: UserCenterComponent,

canActivate: [ XxxGuard ]

children: [ ... ]

}]

练习：创建一个路由守卫： TimeCheckGuard，

作用：如果当前时间是18:00~24:00，允许用户访问user/center，否则禁止访问

|  |
| --- |
| 移动端应用的种类：  **(1)Native App：**  原生App，指使用Java/Kotlin为Android、OC/Swift为iOS开发应用程序，直接运行与手机操作系统上  优势：运行速度快 劣势：两套代码不跨平台，且必须下载  **(2)Web App：**  使用HTML/CSS/JS技术编写类似原生App的应用，代码运行于手机中的浏览器(如WebView)中  优势：无需提前下载、一套代码到处运行 劣势：运行效率低、不能访问手机底层系统服务  **(3)Hybrid App：**  使用HTML/CSS/JS技术编写类似原生App的应用，并混入部分Java/OC等驱动代码以调用系统底层服务，最终运行于操作系统中  优势：结合了前两种的优势  **(4)Dart / Flutter —— 代表着未来** |

## 6.基于Angular的UI组件库

Angular相关的组件库：

(1)Ionic

(2)Material

(3)Zorro

(4)Zorro Mobile

Ionic概述：https://ionicframework.com/

Ionic是一个基于HTML/CSS/JS技术的，创建混合App的UI组件库技术。底层可以不依赖于任何框架(引入.css和.js就可以运行)，也可以与Vue.js、React、Angular框架整合在一起，作为它们的第三方组件库使用。

使用方法：

(1)下载全局脚手架工具

npm i -g ionic

(2)选择一个目录，运行脚手架工具在当前目录下创建一个项目

ionic start 项目名 blank

ionic start 项目名 tabs

ionic start 项目名 sidemenu

创建好项目后会自动调用npm i下载所依赖的第三方模块

(3)进入项目目录，运行它

cd 项目名

npm start

(4)用(手机)浏览器访问项目

http://127.0.0.1:4200

## 课后任务：

(1)复习：整理完整版Angular知识点思维导图

(2)(如果能上网的话)安装Ionic CLI脚手架工具，创建三个不同类型的项目：①空白项目、②页签式导航项目、③侧边菜单项目

(3)查看Ionic项目的目录结构，查找与普通Angular项目的异同

# day06

## 复习：见思维导图

md：Meterial Design Android Studio

ios XCode

HBuilder：可以打包Android/iOS安装程序

npm config set registry https://registry.npm.taobao.org/

## 1.Ionic组件库

第三方提供的一套WEB UI组件库，适用于编写Hybrid App。可以与Angular、Vue、React组合使用，也可以单独直接使用。

官网： ionicframework.com

Ionic CLI = Angular CLI + UIModule + 移动端打包工具....

使用步骤：

(0)安装最新版的Node.js/NPM/NPX

(1)下载最新版本Ionic CLI脚手架工具，并全局安装

npm i -g ionic

(2)运行Ionic CLI脚手架，创建空白项目

ionic start 项目名 blank

ionic start 项目名 tabs

ionic start 项目名 sidemenu

(3)进入项目目录，并启动开发服务器

npm start

(4)启动客户端浏览器，访问开发服务器

http://127.0.0.1:4200

|  |
| --- |
| Ionic的九种主题色——color：  primary(蓝白)、 secondary(青白)、 tertiary(偏紫白)、 danger(红白)、 warn(黄白)、 success(绿白)、 dark(黑白)、 medium(灰白)、 light(亮色) |

## 2.Ionic App的页面布局

<ion-app>

ion-header > ion-toolbar > ion-title > 标题

ion-content[padding] > 内容

ion-footer > ion-toolbar > ion-title > 内容

</ion-app>

--------------------------------------------------------

Ionic主体中的“栅格布局系统”——仿Bootstrap3：

<ion-grid >

<ion-row>

<ion-col no-padding size="宽" offset="偏移" push="右推" pull="左拉" >

</ion-col>

</ion-row>

</ion-grid>

|  |
| --- |
| Module：模块  Model：模型，MVVM中第一个M  Modal：模态对话框——只要不关闭就无法使用其父窗口 |

## 3.Ionic提供预定义组件

Icon：图标 <ion-icon name="">

Button：按钮 <ion-button>

Badge：徽章 <ion-badge>

Card：卡片 ion-card > ion-card-header + ion-card-content

Alert：警告框，必须使用AlertController创建并呈现

Modal：模态框，必须使用ModalController创建并呈现

Item：列表项

List：列表

InfiniteScroll：无穷滚动组件，滚动到底部时加载更多内容

Refresher：刷新器，从顶部显示加载更多内容

## 课后任务：

(1)读取Ionic手册，完成InfiniteScroll和Refresher组件的使用

注意：需要使用http://www.codeboy.com/data/product/list.php提供的跨域数据。

(2)自学Ionic中Slides组件和Tabs组件的使用

# day07

复习：

Ionic是一个适用于移动端混合App的UI组件库，可以与Angular、Vue、React整合使用，也可以完全独立使用。

常用组件：

图标：<ion-icon name="home">

按钮：<ion-button size="" color="">

徽章：<ion-badge color="">

警告框： this.alertController.create({

}).then((d)=>{d.present()})

模态框： this.modalController.create({

}).then((d)=>{d.present()})

弹出框： this.popoverController.create({

}).then((d)=>{d.present()})

列表项： <ion-item detail href="#">

<ion-avatar slot="start"> img </ion-avatar>

<ion-thumbnail> img </ion-thumbnail>

<ion-label></ion-label>

<ion-button slot="end">

</ion-item>

列表： <ion-list></ion-list>

卡片： <ion-card> ion-card-header + ion-card-content</ion-card>

-----------------------------------------------------------

页面基本结构：

<ion-app>

<ion-header> ion-toolbar...

<ion-content padding>...

<ion-footer> ion-toolbar...

</ion-app>

------------------------------------------------------------

网格布局系统：

<ion-grid>

<ion-row>

<ion-col size="3" offset="" pull="" push=""> div </ion-col>

</ion-row>

</ion-grid>

## 1.Ionic常用组件：

无穷滚动：

<ion-infinite-scroll threshold="20px" (ionInfinite)="loadData">

<ion-infinite-scroll-content loadingSpinner="" loadingText="">

</ ion-infinite-scroll>

刷新器：<ion-refresher>

搜索条：<ion-searchbar>

滑动轮播： <ion-slides> <ion-slide></ion-slide> </ion-slides>

标签页： <ion-tabs></ion-tabs>

输入框： <ion-input>

## 2.Angular & Ionic 阶段项目

实现步骤：

(1)创建空白的Ionic项目模板

(2)创建项目必需的组件

ng g component main 盛放页签组件的页面

ng g component index 首页(1F/2F/3F)

ng g component product-list 商品列表页

ng g component cart 购物车页

ng g component login 登录页

ng g component not-found 404页

ng g component product-detail 商品详情页

确保每个组件都在app.module.ts中进行了声明：declarations

(3)修改根组件模板app.component.html，使用MainComponent(其中要创建页签组件)

<ion-app>

<ion-router-outlet></ion-router-outlet>

<app-main></app-main>

<ion-app>

(4)编辑main.component.html，声明页签组件

<ion-tabs>

<ion-tab-bar>

<ion-tab-button tab="路由地址">图标+文字</ion-tab-button>

</ion-tab-bar>

</ion-tabs>

(5)编辑路由配置模块文件app-routing.module.ts，声明并注册路由词典

const routes = [

{path: '', component: IndexComponent},

{path: 'product-list', component: ProductListComponent},

{path: 'product-detail/:pid', component: ProductDetailComponent},

{path: 'cart', component: CartComponent},

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

{path: '\*\*', component: NotFoundComponent},

];

此时可以使用客户端进行测试，访问页签中的每个按钮，可以在路由出口显示对应路由组件页面

-------------------------------------------------------------------------------

(6)开始修改主页组件模板：index.component.html，添加页面内容

<ion-header>

img + ion-searchbar

</ion-header>

<ion-content>

slides + ion-item + ion-grid>ion-card + .....

</ion-content>

模型数据：carouselItems、recommendedITem、topSaleItems、newArrialItem、mySlides、http

--------------------------------------------------------------------------------

(7)修改商品列表页模板：product-list.component.html，添加页面内容

<ion-header>

ion-button + ion-searchbar

</ion-header>

<ion-content>

ion-card>ion-item + ion-infinite-scroll + ion-button

</ion-content>

模型数据：productList、pno、hasMore、http

--------------------------------------------------------------------

(7)修改商品详情页模板：product-detail.component.html，添加页面内容

<ion-header>

ion-button + ion-title + ion-button

</ion-header>

<ion-content>

ion-item + ion-slides + ion-card > ion-item

</ion-content>

模型数据：pid、product、http、mySlides

|  |
| --- |
| 项目运行效果图在FTP上的“Ionic项目运行效果图.zip”压缩包内 |

## 今日任务：

(1)完成商品详情页内容——稍有难点

①从www.codeboy.com获得的商品详情是字符串，其中内容形如：

'<div><img src="img/x.jpg"><img src="img/y.jpg"></div>'

需要使用str.replace方法将其中的图片路径改为codeboy服务器上的地址，变成形如如下的字符串：

'<div><img src="http://www.codeboy.com/img/x.jpg"><img src=" http://www.codeboy.com/img/y.jpg"></div>'

②HTML字符串如果直接使用{{txt}}绑定到模板中，会呈现为innerText而不是innerHTML！使用如下方法可以渲染为innerHTML:

<any [innerHTML]="txt">

(2)根据设计图完成“购物车页面”和“登录页面”——没有难点