复习：

Angular是一个渐进式的MV\*框架，适用于数据操作为主的WEB、移动WEB、桌面应用。

Angular的核心概念：

**(1)模块**：NgModule，抽象概念，用于封装组件、指令、管道等对象，与项目中的“功能模块”的概念对应。

**(2)组件：**Component，是一块可复用的页面区域，包含独立的模板、样式、数据。创建组件的步骤：

1)创建组件 cc.ts

@Component({selector: 'cc', template: ''})

export class CartCounter{ }

2)注册组件 app.module.ts

@NgModule({

declarations: [CartCounter]

})

export class AppModule{ }

3)使用组件 app.component.ts/html

<cc></cc>

**(3)数据绑定**

1)组件->DOM： {{ msg }}

2)组件->DOM(属性绑定)：<img [src]="imgUrl">

3)DOM->组件(事件绑定)：<a (click)="doClick()">

4)组件<->DOM(双向绑定)：<input [(ngModel)]="kw">

**(4)指令**

1)组件

2)结构型指令：影响DOM结构，必须以\*开头

\*ngIf、\*ngFor、\*ngSwitchCase、\*ngSwitchDefault

3)属性型指令：只影响元素的属性，必须用[]括起来

[ngClass]、[ngStyle]、[ngSwitch]、[(ngModel)]

(5)过滤器/管道

(6)装饰器和元数据

(7)服务和依赖注入

(8)路由

|  |
| --- |
| 创建Angular常用对象的快捷方式——使用全局的ng命令：  (1)创建组件  ng generate component 组件名  ng g component 组件名 |

1.Angular中的指令

1)组件

2)结构型指令：影响DOM结构，必须以\*开头

\*ngIf、\*ngFor、\*ngSwitchCase、\*ngSwitchDefault

3)属性型指令：只影响元素的属性，必须用[]括起来

[ngClass]、[ngStyle]、[ngSwitch]、[(ngModel)]

**ngModel指令的使用：**

1)在主模块中导入FormsModule模块，否则会提示ngModel未知

@NgModule({ //app.module.ts

imports: [ FormsModule ]

})

2)在组件模块中进行双向数据绑定

<input [(ngModel)="模型变量"]

|  |
| --- |
| 如何使用JS监视模型数据的改变：  **Vue.js: 使用watch属性**  { data:{kw:''}, watch:{ kw(){ } }}  **Angular： 使用set访问器**  exports class MyDemo{  \_kw = '';  get kw(){ return this.\_kw; }  set kw(val){ this.\_kw = val; }  } |

午间练习：实现“勾选我同意则显示提交按钮，否则隐藏提交按钮”

|  |
| --- |
| Vue.directive('focus', {  inserted( el ){  el.focus()  }  }) |
| <input v-focus> |

2.自定义指令

**(1)创建指令**

@Direrective({ selector: '[xzFocus]'})

export class XzFoucs{

constructor( el: ElementRef ){

el.nativeElement就是原生DOM对象

}

}

**(2)注册指令**

@NgModule({

declarations: [ XzFocus]

})

**(3)使用指令**

<input xzFocus >

练习：创建自定义指令 xzFlash，凡是声明了该指令的元素，每隔200ms就闪动一次

|  |
| --- |
| Vue.js中没有预定义过滤器  Vue.js自定义过滤器：  Vue.filter('sex', function(val){  return ....;  })  {{ tomSex | sex }} |
| Angular提供了预定义过滤器；  也允许用户自定义过滤器； |
| Angular.js 1.x中，过滤器写作： filter  Angular 2.x中，过滤器写作： pipe |

3.Angular预定义过滤器/管道

**(1)date：把对象或数字转换为特定格式的日期字符串**

{{pubTime | date: 'yyyy-MM-dd HH:mm:ss' }}

**(2)number：把数字显示为指定整数位和小数位的字符串**

{{score | number:'8.1-3' }}

**(3)currency：把数字显示为货币形式，逗号+两位小数+货币符号**

{{productPrice | currency:'￥' }}

**(4)lowercase：把字符串转换为纯小写形式**

**(5)uppercase：把字符串转换为纯大写形式**

**(6)titlecase：把字符串转换为每个单词首字母大写形式**

**(7)slice：输出数组或字符串中一部分**

{{msg | slice:2:8}}

**(8)json：把对象转换为JSON字符串格式**

4.Angular中自定义管道

(1)创建管道

@Pipe({ name: 'sex' })

export class SexPipe{

transform(val){ return ...; }

}

(2)注册管道

@NgModule({ declarations: [SexPipe] })

(3)使用管道

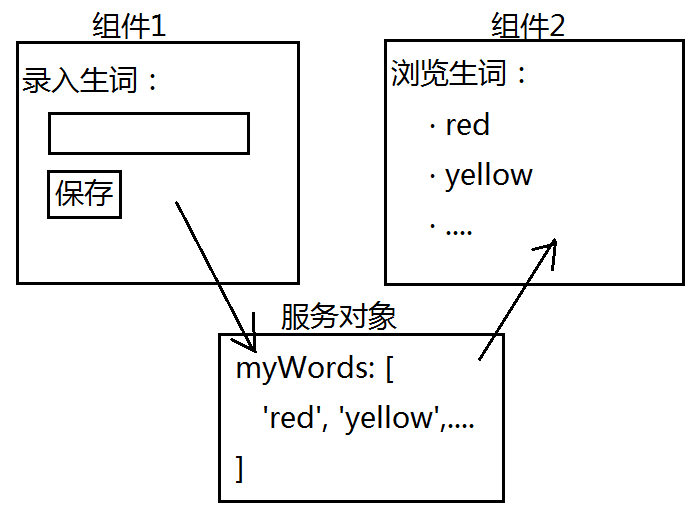
{{teacherSex | sex }}

|  |
| --- |
| var ProductList = {  data(){return { plist: [] }}, //维护DOM 接待顾客  mounted(){  axios.get(url).then((res)=>{this.plist=res.data}) //存取数据 厨师做饭  }  } |

5.Angular中的服务对象(Service) —— 需要反复多次深入理解

**Component**：负责维护DOM，并提供DOM操作需要的方法——前台接待人员

**Service**：负责为组件提供底层支持，如日志服务、计时服务、存储服务等——后厨工作人员



(1)创建服务对象

@Injectable( )

export class WordsService{ }

(2)注册服务对象

@NgModule({ providers: [WordsService] })

(3)使用服务对象

class XxComponent{

ws:WordsService = null; //组件依赖于服务对象

constructor( s: WordsService ){ //将服务注入给组件

this.ws = s;

}

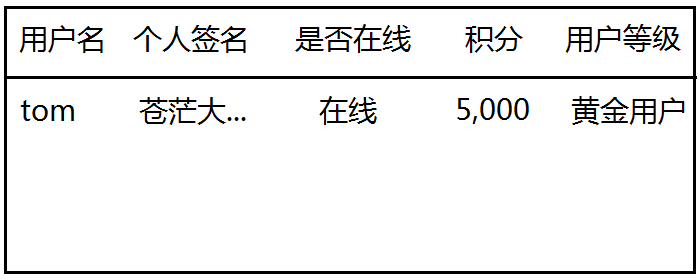
}

课后练习：组件/管道练习

(1)创建自定义组件UserListComponent，其中包含数据：

ulist: [ {uname: 'toM', signiture:'...', isOnline: true, score:5000},...]

把这些数据绑定在一个表格中显示：



提示：需要自定义管道：online、userlevel

(2)创建自定义组件TodoList，实现“待办事项列表”——不需要使用组件间通信

(3)初步理解服务对象，完成“保存生词”和“显示生词”练习