复习：

见思维导图

1.服务 和 DI

Component：组件，负责维护DOM/视图——相当于服务员。

Service：用于为组件提供底层支持，如数据存储、日志服务、定时器服务等等——相当于厨师。注意：组件依赖于服务，使用DI注入给组件即可。—— 这样写符合很多的“软件设计原则”。

**(1)创建服务**

@Injectable( )

export class LogService{ }

**(2)注册服务**

@NgModule({

providers: [ LogService ]

})

**(3)使用服务 —— 不允许服务操作DOM视图，必须注入给组件**

XxComponent{

xs = null;

constructor(s:LogService){ //依赖注入

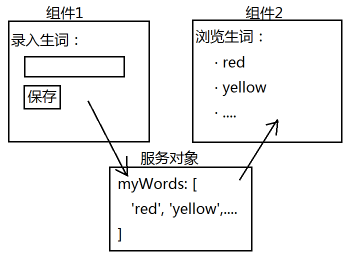
this.xs = s;

}

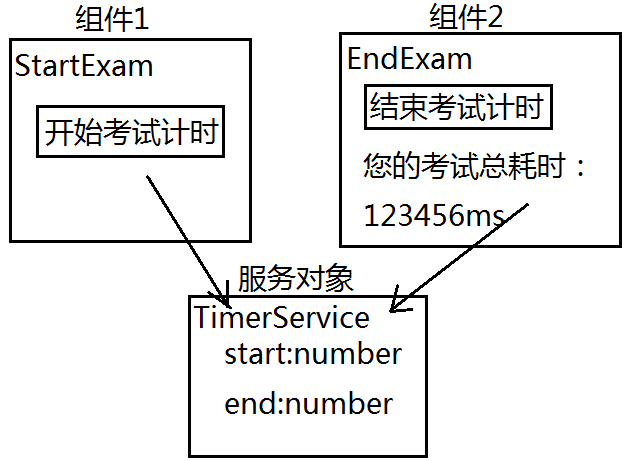
}

|  |
| --- |
| **重要理论：依赖注入（Dependency Injection，简称DI）**  Dependency：依赖，如果创建A对象必须先有B对象，就说A依赖于B，如“Car依赖于Moto”。依赖对象如何解决：  **(1)穷人式思维： 手工创建每一个依赖的对象**  functioin 开车(){  var c1 = new Car( new Moto(), new Sofa( new Skin(),new Thread()... ) );  c1.run( );  }  **(2)富人式思维：只需要声明自己依赖的对象类型，其*所在环境*(Angular提供)会自动创建出需要的对象来，并传递进来 —— 称为“注入(Inject)”**  function 开车( c2: Car ){  c2.run( );  } |

练习：创建组件：word-save和word-display，负责保存生词和显示生词。创建服务：WordService，负责存储和读取所有的生词。



练习：创建组件StartExam，点击“开始考试计时”记录当前系统时间；创建组件EndExam，点击“结束考试计时”记录当前系统时间，并获取结束时间与开始时间的差值，显示在视图中——注意：计时服务应该使用一个Service



2.软件工程：软件设计原则

(1)KISS：Keep It Simple & Stupid

代码越简单越傻瓜越好

(2)DRY：Don't Repeat Yourself

不要重复自己写过的代码，尽量复用代码

(3)SRP：Single Responsibility Principle

单一责任原则

(4)OCP：Open Close Principle

开闭原则，对外部扩展持开放态度，对内部修改持闭合态度

(5)HCLC：High Cohesion，Low Coupling

高内聚低耦合原则

(6)LoD：Law of Demeter

迪米特法则，也称为“最少知识原则”，每个对象知道的数据/操作越少越好

3.扩展知识：TypeScript语法基础

ES是ECMA委员会推出的JS行业标准语言；

TS是Microsoft推出的兼容全部JS特性的新语言，是ES的超集。

注意：浏览器只能执行ES，所有的TS必须转换为ES语法才能被执行 —— 称为编译。Angular脚手架项目自带TS编译器，会在服务器端把所有的TS代码编译为ES，供客户端下载使用。

TS比ES增加的三项主要内容：

**(1)强制类型声明**

ES是弱类型语言，TS是强类型语言——所有的属性、形参、函数都需要声明数据类型，如string、number、boolean、object、any、xx[ ]

**(2)访问控制修饰符**

private：私有成员，只能在当前class内部使用

protected：被保护成员，只能在当前class及子class内部使用

public：共有成员，可以被其它class直接使用

提示：一种特殊的简写方法

|  |  |
| --- | --- |
| class MyClass {  private age: number = null;  constructor(age:number){  this.age = age;  }  } | class MyClass {  constructor(private age:number){  }  } |

**(3)interface接口**

**class**：类，是对象的模板，可以实例化出多个对象；每个class中可以包含多个 成员属性 + 成员方法。

**interface**：接口，是一种特殊的“class”，不能实例化；不能包含成员属性；接口中的方法不能有方法体；接口不能被class继承 —— 接口只能被类“实现(implements)”—— 类实现接口时必须实现接口能声明的所有未实现的方法。

结论：接口用于规范要求类必须提供特定的方法！

|  |
| --- |
| Vue.js中组件生命周期钩子函数：  {  beforeCreate(){},  created(){},  beforeMount(){},  **mounted(){},**  ...  } |

4.Angular中组件的生命周期钩子函数

@Component({...})

export class MyHeader implements OnInit{

ngOnInit(){ } //每个钩子函数都有一个对应的接口

}

代码中的OnInit接口要求MyHeader必须提供ngOnInit()生命周期钩子函数。

提示：更多的生命周期钩子函数请参加https://angular.cn/guide/lifecycle-hooks

课后任务：使用Angular组件和服务重构“学子商城”前台系统

(1)创建组件UserRegister，模板效果如http://www.codeboy.com/register.html

(2)创建组件UserLogin，模板效果如http://www.codeboy.com/login.html

(3)创建服务UserService，提供服务方法register()和login()，接收组件传递的数据，实现用户注册和登录。

提示：先创建必需的对象，模拟实现用户注册和登录；再**自学使用**Angular提供http工具实现异步PHP请求，可参看手册：https://angular.cn/guide/http