# 0. AngularJS介绍

从课程的第三阶段开始，我们就开始学习前端框架，尤其是第四阶段。既然提到前端框架，我们回顾一下，都学了哪些？jquery、two.js、chart.js、bootstrap。对于框架，大家的定义是什么？优秀的可重复使用的代码，单独提取并封装，供重复使用，完成复杂功能，提高开发速度，提高代码质量，并保证良好的浏览器兼容性。这对于解决繁杂的原生代码开发，是一件非常棒的事情。但是，这同时意味着，即使课程里学了很多框架，依然会有很多等待着大家的探索。因此，学习框架，不再主要是记忆，而是理解、动手查阅和编写的能力。

因此，我们为期5天的课程，3天是知识点的讲解，讲解框架本身，还有两天是项目。那我们首先谈谈 AngularJS。

AngularJS  诞生于2009年，由Misko Hevery 等人创建，后为Google所收购，是一款优秀的前端JS框架，已经被用于Google的多款产品当中。

AngularJS是一款基于 MVC 设计模式的 JS 框架。言下之意，是基于 JS 进行封装的，但是，什么是 MVC？什么是设计模式？这些涉及到软件工程理论的一些知识。、

因此，首先，我们先讲讲软件工程中的设计原则和模式。

# 1.设计原则和模式

## 1.1. 概述

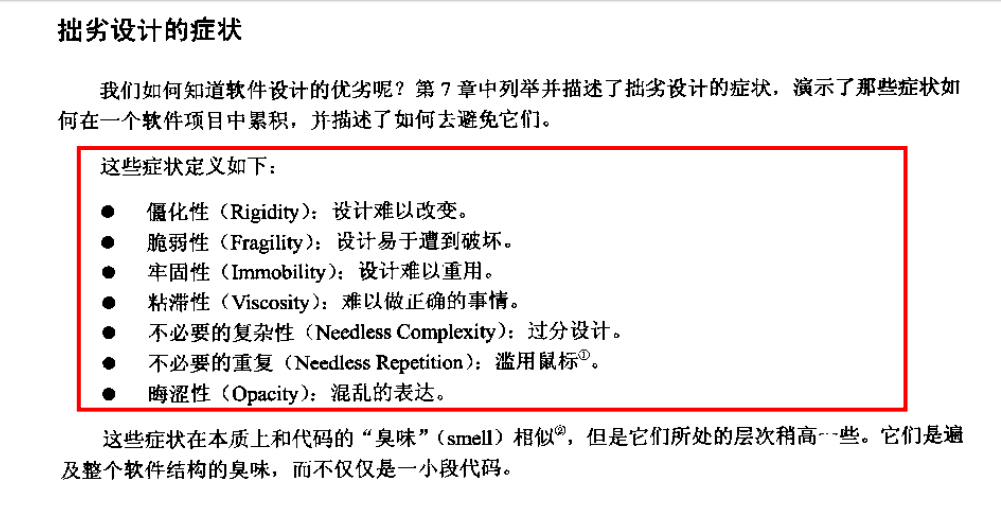
### 1.1.1. 【概述】概述

一个工具或者语言，用久了就可以熟能生巧，就想砌墙的工人一样长年累月地砌墙，他也知道如何把墙砌整齐，如果多块好省地干活，是一个人的本能。我们html、js、css用的很顺手，这非常好，但这只是一个程序员应该具备的基本能力！于是我们被冠于一个外号“代码民工”--软件行业的体力劳动者。

但是如果你通晓了设计原则和设计模式就不同了，你可以站在一个更高的层析去欣赏程序代码、软件设计、架构，完成从代码民工到架构师的蜕变。这里的通晓不是记住各种设计模式的含义、优缺点，没有工作经验的积累是不可能真正理解设计模式的。所以在设计原则、模式的这一块，对大家的要求是“了解”。（学习分为3个层析：了解、明确、掌握）

好了接下来，举一个反面的例子，也是新手没有基于基本的设计原则和设计模式编码，我们一起来看一下。

### 1.1.2. 【概述】拙劣设计的特点



* 僵化性

难以对软件进行改动，即使是简单的改动。如果单一的改动会导致有依赖关系的模块中的连锁改动，那么设计是僵化的。必须改动的模块越多，设计就越僵化。

* 脆弱性

在进行一个改动时，程序的许多地方就可能出现问题，从而使开发团队就想一直不停追逐自己尾巴的狗一样。

* 牢固性

设计中包含了其他对系统有用的部分，但是要把这些部分从系统中分离出来所需要的努力和风险是巨大的

* 粘滞性

当需要改动代码时，开发人员可以保持当前的设计风格，也可以破坏设计风格，当保持设计风格比破坏设计风格时，就表明设计具有较高的粘滞性

* 不必要的复杂性

如果设计中包含当前没有用的组成部分，它就含有不必须要的复杂性；比如说过分的为未来潜在变化预留代码。

* 晦涩性

晦涩性是指模块难以理解。代码可以清晰、富有表现力的方式编写，或者可以用晦涩、费解的方式编写。代码随着时间而演化，往往会变得越来越晦涩。为了使代码的晦涩性保持最低，就需要持续保持代码清洗并富有表现力。

上述特点都是在进行代码的编写、迭代时会出现，也就是在应对不断变化的需求时会出现，你说这是谁的错？是开发人员你的错，那么怎样才能实现拥抱变化而保持代码简洁明了呢？我们需要遵循软件界的前辈们总结一套可以反复使用的经验，它可以提高代码的可重用性，增强系统的可维护性，以及解决一系列复杂的问题。所谓的经验就是前辈们总结的设计模式。那接下，我们来认识一下这些设计原则和模式。

## 1.2. 设计原则和模式

前辈们总结的设计模式给了我们指导，这些专家们提出了6大设计原则，但是这6大设计原则仅仅是一系列的“口号”，真正付诸实施时还需要详尽的指导方法，于是23种设计模式出现了。

### 1.2.1. 【设计原则和模式】软件设计原则

(1)YAGNI——不写不需要的代码

You aren’t gonna need it

只需要将应用程序必须的功能包含进来，而不要试图添加任何其他你认为可能需要的功能

(2)KISS——代码越简单越好，比如语义化的标记、减少HTML中<div>的嵌套

Keep It Simple And Stupid

保持简单和一目了然（傻瓜式），代码要简单，把复杂的逻辑用简单的代码展示出来，

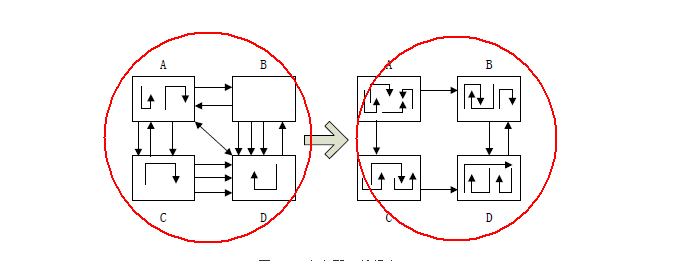
一目了然，加上必要的注释，对函数、变量合理命名，并将代码合理组织。

(3)DRY——不要Ctrl+C/V，使用封装和重用

Don’t Repeat Yourself

不要重复自己，换言之，就是不写重复的代码，如果你的代码中出现大量的重复代码，那么就是时候来将重复的代码进行封装从而实现代码的复用

(4)High Cohesion，Low Coupling---高内聚，低耦合。把各个对象的功能封装在自身对象，对象之间可以相互调用，但是不用彼此包含。比如：尽量不要在对象内部new对象，可以传入。内聚：一个组件内的不同部分间的关系；耦合：两个组件间的关系



(5)SRP——单一责任原则。

Single Responsibility Principle



举例：

<div class=”col-xs-6 col-md-3 alert text-right”>

//这一个div即实现布局，又实现文字方向调整，违背可SRP，应该这样：

<div class=’col-xs-6 col-md-3’>

<div class=’text-right’>

</div>

</div>

(6)OCP——对扩展开放，对于修改关闭。

Open Closed Principle



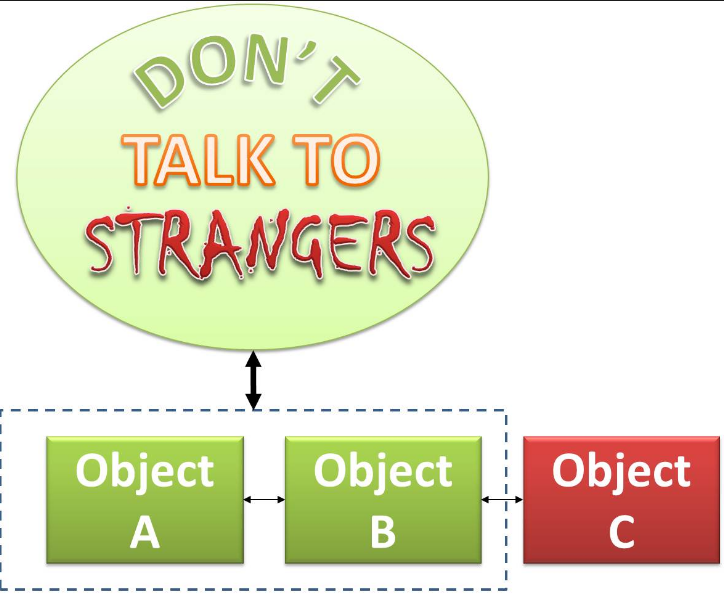
Open Chest Surgery is Not Needed When Putting On A Coat,穿衣服时无需做个开胸手术。即，代码可供别人进一步使用，但不能被别人修改。

比如JS代码中的方法参数以及返回值的定义，使用arguments的定义。

比如，JS中的构造函数的定义，扩展实例属性。

(7)LCP迪米特法则/最少知识法则——封装的功能单位要小，只与自己相关。

Least Knowledge Principle



多个原则之间可能会有冲突，需要灵活选择。

比如：

function Driver()

{

var car = new Car();

car.start();

car.stop();

}

代码这么写肯定是有问题的，因为司机是不需要造车的，对司机来说，造车是陌生的（stranger），这时候应该这么写：

function Driver(car)//司机只需要知道如何开车，不需要知道如何造车

{

car.start();

car.stop();

}

### 1.2.2. 【设计原则和模式】软件设计模式

优秀的项目经验的总结，用于某种特定的场景下的特定解决方案：如何设计代码。这种称为设计模式。一共23种 + MVC 模式。

### 1.2.3. 【设计原则和模式】MVC

MVC ：M为model（模型）、V为View（视图）、C为controller（控制器）。

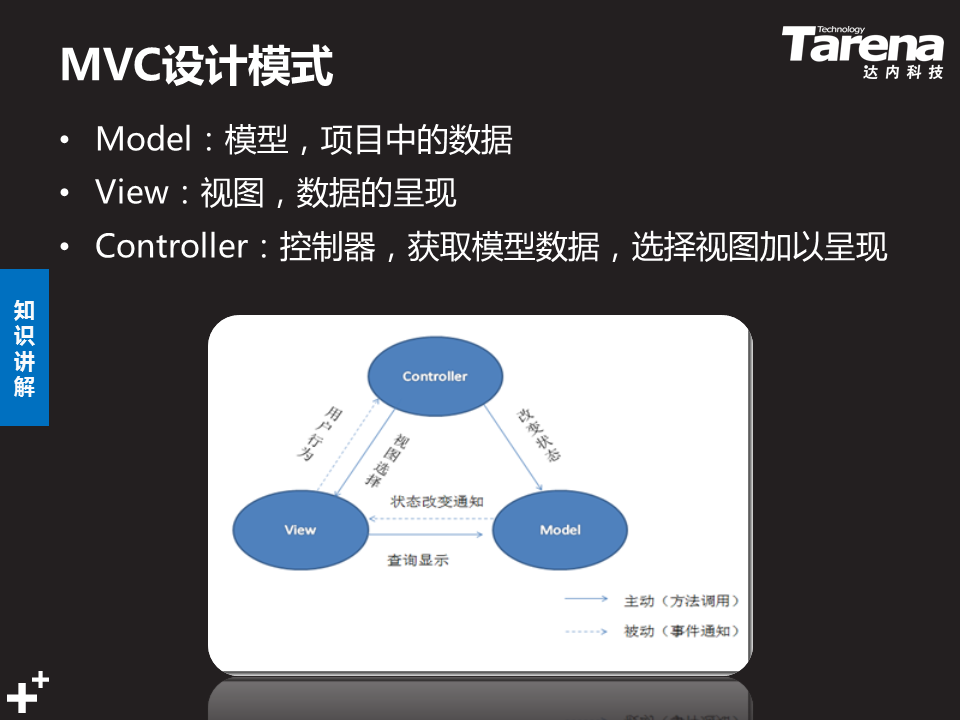
M：业务数据，数据的模型，数据的封装定义。比如商品、订单、用户、评论等。每一种数据是一种数据模型，在js中，各种数据类型的变量用于表示数据模型。

V：视图，表示业务数据的表现，由HTML来表示。需要加强的 HTML，比如多个产品显示的是相同的，但是不能大量的拷贝和复制，因此需要加强的 HTML。

C：控制器：负责控制数据的获取、删除和更新。由js中的各种function来实现。

比如，需要从服务器端获取三个商品，然后显示在界面上。那么，需要方法获取数据，需要回调方法显示数据。这个过程中，需要定义数据的格式，需要定义数据的显示方式。

原生代码实现时，整个过程都得自己写，总是会疑惑于思路。那么，有了基于 MVC 的模式，以后会形成一种固定方式：先定义数据模型，定义显示方式，由控制器来控制它们。三者之间的关系见PPT上的图。



# 2. AngularJS概述

## 2.1. AngularJS概述

### 2.1.1. 【AngularJS概述】什么是 AngularJS

诞生于09年，后被Google收购，是一个JS框架，适用于以数据操作为主的 SPA（single page application）应用，实现数据的增删改查等操作。

### 2.1.2. 【AngularJS概述】四大特征

四大特性：使用 MVC 模式（不再使用原生dom操作、jquery操作），双向数据绑定，依赖注入，模块化设计。

jQuery和AngularJS的关系：jQuery是对于DOM操作的进一步封装，AngularJS是对于jQuery的进一步封装，基于jQuery，自带一个 jQLite（一个轻量级的jQuery，只包含DOM操作的各种函数，不包含动画等内容）。

## 2.2. 第一个AngularJS

### 2.2.1. 认识AngularJS-1.4.1压缩包

**介绍压缩包：**

* Css：样式文件
* Js中的i18n：internationalization，共18个字母，表示项目国际化，用于根据客户端语言环境的不同，呈现出不同的语言。其中，包含不同的语言包，目前，主要是英文和中文。
* JS：其中的文件名和手册中的分类是对应的，手册中的ng就是angularjs的简称。每个章节就是一个模块，对应一个js文件，实现特定的功能。实现何种功能，就导入何种 js 文件。
* 本课程中主要讲解 angular，也是最核心的模块。该文件中，大部分使用匿名函数，不会使用全局对象和全局命名空间。
* 如果需要添加样式，则可以将bootstrap.css 的样式文件，实现多个框架的无缝整合。

查看 angularjs的源码，会发现，整个框架只有一个污染点： window.angular，其中的对象以$开头，如$http、$interval、$window、$timeout、$location；ng内部专用的对象(禁止程序员调用的对象)都以$$开头。

### 2.2.2. 第一个AngularJS

打开webstorm，新建工程，创建js\css\img文件夹

**案例：**新建1.html，引入angularJS文件，写一个h标签，看看页面发生的变化

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<h1>第一个AngularJS页面</h1>

<script src="js/angular.js"></script>

</body>

</html>

何为增强的 HTML 代码？比如需要在页面上显示一些计算结果？那么需要首先，定义表达式，然后使用相应的指令来计算它。

# 3. 表达式和指令

AngularJS表达式由数据变量和运算符构成，可用于在HTML中执行运算

语法：{{表达式}}

作用：在当前位置“输出”表达式的值

在1.html中添加以下代码：

<p>{{1+2}}</p>

刷新页面，发现只是{{1+2}},为什么表达式没有成功计算出来？

在AngularJS中有作用范围这么一个概念，在网页中可以指定从哪到哪是作用范围，在作用范围内就可以被AngularJS处理，这个范围叫做AngularJS应用程序。那么如何指定这个应用范围呢？我们需要用到ANgularJS的指令。

接下来表达式和指令是混合学习，为了让学生不晕菜，可以画一张表格。

## 3.1.表达式

### 3.1.1. 【表达式】算术运算

基本算术符是可以的，但是++、--是不可以的。

### 3.1.2. 【表达式】比较预算

各种比较运算符都可以。

### 3.1.3. 【表达式】逻辑运算

各种逻辑运算符是可以的。

### 3.1.4. 【表达式】三目运算

可以使用三目运算。

### 3.1.5. 【表达式】特殊运算

调用对象的属性和方法是可以的。

赋值运算中，=可以，但是+=、-=这些不可以。

在2.html中，进行如下编码：

<!DOCTYPE html>

<html lang="en" ng-app>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<h1>第二个AngularJS页面</h1>

<h2>AngularJS表达式</h2>

<h2>算数运算</h2>

<p>加法1+2={{1+2}}</p>

<p>减法1-2={{1-2}}</p>

<p>乘法1\*2={{1\*2}}</p>

<p>除法1/2={{1/2}}</p>

<p>取余3%2={{3%2}}</p>

<!--<p>自增2++ {{2++}}</p>-->

<!--<p>自减2&#45;&#45; {{2&#45;&#45;}}</p>-->

<span ng-init="num1=20;num2=30"></span>

<!--<p>{{num1++}}</p>-->

<!--<p>{{num2&#45;&#45;}}</p>-->

<p>自加和自减是禁止的</p>

<h2>比较运算</h2>

<p>{{num2>num1}}</p>

<p>3>2 {{3>2}}</p>

<p>3>=2 {{3>=2}}</p>

<p>3<2 {{3<2}}</p>

<p>3<=2 {{3<=2}}</p>

<p>3==2 {{3==2}}</p>

<p>3!=2 {{3!=2}}</p>

<h2>逻辑运算</h2>

<p>true && false = {{true && false}}</p>

<p>true || false ={{true || false}}</p>

<p>! false = {{!false}}</p>

<h2>赋值运算</h2>

<p>{{result=3+2;result\*2}}</p>

<!--<p>3+=2 {{3+=2}}</p>

<p>3-=2 {{3-=2}}</p>

<p>3/=2 {{3/=2}}</p>

<p>3%=2 {{3%=2}}</p>-->

<h2>三目运算</h2>

<p>3>2?'yes':'no' = {{3>2?'yes':'no'}}</p>

<h2>特殊运算</h2>

<!--<p>typeof{{typeof (num1)}}</p>-->

<!--<p>instanceof{{age instanceof String}}</p>-->

<!--特殊运算符都是禁止使用的-->

<h2>调用字符串的成员运算</h2>

<span ng-init="name='Tom'"></span>

<p>name:{{name}}</p>

<p>name.length:{{name.length}}</p>

<p>name uppercase {{name.toUpperCase()}}</p>

<p>{{name.substr(1)}}</p>

<h2>创建新对象(多种语法)</h2>

<p>对象直接量语法：{{ {name:'Mary',age:24} }}</p>

<p>对象直接量语法：{{ {name:'Mary',age:24}.age }}</p>

<span ng-init="car={name:'BWM',price:'5000'}"></span>

<p>{{car}}</p>

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

<!--<p>{{new Object()}}</p>-->

<!--new和var这两种关键字在ng中都是不允许使用的-->

<h2>创建数组对象</h2>

<p>{{ [10,20,30,50] }}}</p>

<span ng-init="list=[20,100,80,90]"></span>

<p>{{list[0]}}</p>

<ul>

<li>{{list[0]}}</li>

<li>{{list[1]}}</li>

<li>{{list[2]}}</li>

<li>{{list[3]}}</li>

</ul>

<script src="js/angular.js"></script>

</body>

</html>

练习：20分钟时间，学生自己来写。

使用ngInit指令创建一个“学生数组”Angular变量，其中包含4个学生，每个学生有stuName,sex,Chinese,math,isOnCampus,每个学生包含以上5个属性，把上面数据使用AngularJS表达式输出在一个Table中。

新建3.html，写练习代码

<!DOCTYPE html>

<html lang="en" ng-app>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<div ng-init="stuList=[{stuName:'lucy',sex:'female',chinese:70,math:80,isOnCampus:true},

{stuName:'lily',sex:'female',chinese:100,math:0,isOnCampus:true},

{stuName:'daxu',sex:'male',chinese:80,math:100,isOnCampus:false},

{stuName:'dongdong',sex:'male',chinese:90,math:90,isOnCampus:false}]">

<table>

<thead>

<tr>

<th>stuName</th>

<th>sex</th>

<th>chinese</th>

<th>math</th>

<th>isOnCampus</th>

</tr>

</thead>

<tbody>

<tr>

<td>{{stuList[0].stuName}}</td>

<td>{{stuList[0].sex}}</td>

<td>{{stuList[0].chinese}}</td>

<td>{{stuList[0].math}}</td>

<td>{{stuList[0].isOnCampus}}</td>

</tr>

<tr>

<td>{{stuList[1].stuName}}</td>

<td>{{stuList[1].sex}}</td>

<td>{{stuList[1].chinese}}</td>

<td>{{stuList[1].math}}</td>

<td>{{stuList[1].isOnCampus}}</td>

</tr>

</tbody>

</table>

</div>

<script src="js/angular.js"></script>

</body>

</html>

这个时候抛出“大括号”问题，怎么解决？在chrome中预览3.html页面，然后查看源代码，浏览器在解析DOM树时，对于ng开头的指令或者属性都不认识，所以都不进行操作，直到加载到script中引入AngularJS时，才执行ng应用

有些同学说把AngularJS的引入放在最上边可以不可以？可以轻微改善，但是无法从根本上解决这个问题，怎么根除呢？（引入ng-bind指令）

## 3.2. 指令

是由AngularJS提供的，可以用在HTML中的一些扩展属性、扩展标签。

属性：<div ng-XX=’’>ng是AngularJS的缩写，ng开头的都是AngularJS的属性

标签：以后我们还会学习<ngView></ngView>，这个不是h5的标签，是AngularJS自定义的标签

### 3.2.1. 【指令】ngApp指令

使用这个指令自动载入启动一个AngularJS应用。ngApp指令设计为应用的根元素，一般把它放在页面的根元素附近，例如<body>或<html>标签。

用于指定一个AngularJS应用的范围

每个HTML文档只有一个AngularJS应用能被自动载入启动。文档中找到的第一个ngApp将被用于定义自动载入启动的应用的根元素。要在一个HTML文档中运行多个应用，你必须使用angular.bootstrap来手工启动。AngularJS应用间不能嵌套。

修改1.html为：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<h1>第一个AngularJS页面</h1>

<h2>AngularJS表达式</h2>

<div ng-app>

<p>1+2={{1+2}}</p>

</div>

<script src="js/angular.js"></script>

</body>

</html

如果在div下边加上{{3+2}}这么一个表达式，再看效果，发现是没有执行的

<div ng-app>

<p>1+2={{1+2}}</p>

</div>

<p>3+2={{3+2}}</p>

出了ng-app的作用范围，ng表达式就不好使了。

此时继续调整代码，在{{3+2}}的表达式外边也加一个div并指定ng-app,发现依然是没有效果的，是因为每个HTML文档只有一个AngularJS应用能被自动载入启动。

那么有些同学就会纳闷，如果在html中的不同地方执行AngularJS指令，怎么办？

其实很简单，把ng-app指令放在body或者html中就可以了。

最终1.html代码如下：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<h1>第一个AngularJS页面</h1>

<h2>AngularJS表达式</h2>

<div ng-app>

<p>1+2={{1+2}}</p>

</div>

<p>3+2={{3+2}}</p>

<div ng-app>

<p>3+2={{3+2}}</p>

</div>

<script src="js/angular.js"></script>

</body>

</html>

接下来，复制1.html到2.html，并开始AngularJS表达式接下来的学习。

### 3.2.2. 【指令】ngInit指令

Ng-init 指令允许声明变量，需要注意：为双向绑定。

用法：<ANY ng-init=”变量名=值；变量名=值;…”></ANY>

提示：ng-init声明变量时，无需写var关键字，多个变量赋值表达式用分号分开。

<div ng-init="name='mary';age=18" class="container">  
<h3>测试表达式</h3>  
<p>{{name + age}}</p>  
</div>

### 3.2.3. 【指令】ngBind指令

将制定的表达式的值输出为当前元素的innerHTML

用法：<ANY ng-bind=”表达式”></ANY>

在3.html中，加入以下内容：

<p ng-bind="’姓名’+stuList[0].stuName"> </p>

刷新页面，发现p元素内容刚开始为空，后来再显示内容（如果p内本来有元素，刷新时会显示出来，然后被替换掉）

用途：可以用于代替{{}}语法，输出一个表达式的值，但是可以防止闪动的问题。使用大括号时，先显示大括号，然后更换，所以会有闪动；而使用bind指令是，先没有输出，后显示输出，所以没有闪动。

注意：ng-bind会将表达式的值计算出来，并替换当前元素的innerHTML，已有的内容会被替换掉

<h1 ng-bind="name">原始内容</h1>  
<h1 ng-bind="age">原始内容</h1>

复盘3.html，我们复制粘贴了好几次，违背了一个原则DRY（Don’t Repeat Yourself）

那么ng中有没有循环功能？带着学生去看手册，发现ng-repeat指令。

### 3.2.4. 【指令】ngRepeat指令

为HTML添加循环功能

语法1：<ANY ng-repeat=”临时变量名 in 对象或者数组”></ANY>

语法2：<ANY ng-repeat=”(下标变量名,值变量名) in 对象或者数组”></ANY>

用途：对于指定的对象或者数组中的每个成员，依次赋值给指定的临时变量，将当前的HTML元素循环输出一遍。

ngRepeat指令为集合中的每项实例化一个模板。每个模板的实例拥有自己的域，使用循环变量指向当前集合项上，$index指向当前项的索引或键值。

新建4.html

<!DOCTYPE html>

<html lang="en" ng-app>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="css/bootstrap.css"/>

<title></title>

</head>

<body>

<h1>为HTML添加循环和选择功能</h1>

<h2>循环功能</h2>

<p ng-init="scoreList=[100,20,88,59]"></p>

<ul>

<!--<li ng-repeat="score in scoreList">&#45;&#45;&#45;&#45;</li>-->

<!-- 上面是用来演示循环功能，li变成了4个-->

<li ng-repeat="num in scoreList">{{num}}</li>

</ul>

<!--如果有同学提问：把ng-reapt放在ul中可以吗？不可以！强调ng-repeat会重复当前元素，如果

如果放在ul中，重复ul是没有意义的；演示一下,发现效果也能出来，但是审查元素会发现多了很多UL

-->

<ul ng-repeat="num in scoreList">

<li>{{num}}</li>

</ul>

<!--当然我们也不是非得在列表中使用，比如下边这样-->

<div ng-repeat="num in scoreList">

<span>{{num}}</span>

</div>

</div>

<script src="js/angular.js"></script>

</body>

</html>

### 3.2.5. 【指令】ngIf指令

为HTML添加选择功能

语法：<ANY ng-if=”表达式”></ANY>

用途：当表达式为true时，当前元素会挂到DOM树上，否则会删除

ngIf指令基于{表达式}来移除或重建DOM树的一部分。如果赋给 ngIf的表达式计算结果为假，元素会从DOM移除，其它情况会复制一份元素插入到DOM中。

在4.html中加入以下代码：

< <h2>为HTML添加选择功能</h2>

<span ng-init="isMember=true;">`</span>

<div class="alert alert-success" ng-if="isMember">

会员专区：仅会员可见

</div>

***练习：要求使用ngRepeat循环输出tr、td，仅显示语文分数大于50分的数据***

***20分钟时间。***

***新建5.HTML***

<!DOCTYPE html>

<html lang="en" ng-app>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="css/bootstrap.css"/>

<title></title>

</head>

<body>

<div ng-init="stuList=[{stuName:'lucy',sex:'female',chinese:70,math:80,isOnCampus:true},

{stuName:'lily',sex:'female',chinese:100,math:0,isOnCampus:true},

{stuName:'daxu',sex:'male',chinese:80,math:100,isOnCampus:false},

{stuName:'dongdong',sex:'male',chinese:59,math:90,isOnCampus:false}]">

<table class="table">

<thead>

<tr>

<th>index</th>

<th>stuName</th>

<th>sex</th>

<th>chinese</th>

<th>math</th>

<th>isOnCampus</th>

</tr>

</thead>

<tbody>

<!-- 显示方案1-->

<tr ng-repeat="(index,obj) in stuList" ng-if="obj.chinese > 60">

<td>{{index+1}}</td>

<td>{{stuList[index].stuName}}</td>

<td>{{stuList[index].sex}}</td>

<td> {{obj.chinese}}</td>

<td>{{stuList[index].math}}</td>

<td>{{stuList[index].isOnCampus}}</td>

</tr>

<!-- 显示方案2-->

<tr ng-repeat="(index,stu) in stuList" ng-if="stu.chinese>60">

<td>{{index+1}}</td>

<td ng-repeat="v in stu">{{v}}</td>

</tr>

</tbody>

</table>

</div>

<script src="js/angular.js"></script>

</body>

</html>

总结：创建AngularJS变量的两种方法

* 方法1：使用ngInit指令

不足：把数据和展示混杂在一起！ngInit声明的数据中不能使用New。

* 方法2：使用控制器来创建数据变量

不足： 没有！

面试题：

AngularJS的四大特性

* 采用MVC设计模式（今天只讲这一个）
* 双向数据绑定
* 依赖注入
* 模块化设计

分析：此时，使用ngInit指令，可以定义数据，但是，这种定义方式，显示和数据完全混在在一起，很难重用和维护。需要将数据和显示分开，需要使用 MVC 的模式。

# 4. MVC模型

## 4.1.MVC模型

Model：模型，表示数据，在AngularJS中由JS变量担当

View：视图，表示HTML加上指令，用于显示数据，在AngularJS中由HTML来担当

Controller：控制器，负责操作数据（增删改查），在AngularJS中由function来担当，由于function众多，因此需要使用模块进行管理。官方自带的模块参见说明文档，也可以自定义模块。

使用AngularJS中的MVC模型的步骤：

* + 声明自定义的模块：angular.module(‘模块名’，[依赖列表])
  + 注册模块：将自定义的模块注册给当前应用，如：<html ng-app=”模块名”>
  + 声明控制器：在模块中声明控制器函数，如： module.controller('控制器名’,function(){…})
  + 调用控制器：调用控制器的函数，创建控制器对象，如：<ANY ng-controller=”控制器名称” />
  + 操作：在控制器中增删改查Model数据：$scope.变量名 = 值；
  + 显示：在View中呈现model数据 ：{{变量名}}

**新建6.html**

<!DOCTYPE html>

<html lang="en" ng-app="myModule">

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="css/bootstrap.css"/>

<title></title>

</head>

<body>

<h1>创建Model数据的两种方法</h1>

<h2>方法1：使用ng-init指令来创建Model数据</h2>

<h2>方法2：使用controller来创建Model数据</h2>

<!--ng-controller调用几次，控制器对应的函数就会执行几次-->

<!--<div ng-controller="myController">-->

<!--</div>-->

<div ng-controller="myController">

<p>{{emp}}</p>

<p>{{emp.age}}</p>

<p>list is {{empList}}</p>

</div>

<script src="js/angular.js"></script>

<script>

var app = angular.module('myModule', []);//声明一个自定义模块

app.controller('myController', function ($scope) {//在模块中声明一个控制器

console.log('一个控制器对象被创建了');

var num = 100;//这样只是声明了一个函数内的局部变量

// $scope声明的是模型数据

$scope.emp = new Object();

$scope.emp.name='TomCruise';

$scope.emp.age=30;

$scope.emp.birthday = new Date();

$scope.empList = new Array();

$scope.empList.push({name:'zzl',age:28,birthday:new Date()});

$scope.empList.push(new Object());

})

</script>

</body>

</html>

由案例总结MVC的创建使用步骤为：

(1) 创建一个模块，使用ngApp指令确定该模块的作用范围：注册完毕后，使用控制台检查一下是否有错误；

(2) 在模块中声明一个控制器组件，使用ngController指令声明控制器实例；

(3) 为控制器注入$scope对象，声明模型数据，将数据声明为$scope的属性；

(4) 在视图中绑定模型数据，进行呈现，使用{{}}的方式。

### 4.1.1. 【MVC模型】Model

为了呈现数据，必须先声明 model。创建model时，可以使用ngInit指令，但是，这种情况下，数据和显示混在在一起，因此，建议使用controller来声明模型数据。

### 4.1.2. 【MVC模型】View

View 就是使用各种表达式和指令来显示内容。

### 4.1.3. 【MVC模型】Controller

老版本的angularjs可以直接写原生的function作为controller使用，但是新版本废弃了这种写法，是为了避免污染。

新版本中，将各种控制器按照类别和功能进行分类，即按照模块进行划分，即模块化设计。

**案例讲解：使用test3\_使用 MVC 模型.html讲解创建的过程，使用控制台查看错误。可以创建多个控制器，并分别使用数据模型。**

### 4.1.4. 【MVC模型】使用控制器创建模型数据

需要注意：

1. 数据的范围：位于各自的控制器范围里，由声明控制器的<html>标签来决定范围。
2. 控制器的范围不易过大，对应于页面上的一个组成部分。

**案例：见test4\_使用 MVC 模型展示复杂数据.html。**

## 4.2.其他指令

### 4.2.1. 【其他指令】显示指令

讲解其他和显示相关的指令。

1、ngSrc指令：Angular的{{hash}}等标记在src属性中不能正确工作：浏览器会使用包含{{hash}} 的原始文本的URL，在Angular更新{{hash}}内的表达式之前。 ngSrc 指令解决了这个问题：img等标签的src属性中包含{{}}时，会产生404错误。

2、ngShow 和 ngHide

根据ngShow或者ngHide属性上的表达式来显示或隐藏给定的HTML元素。元素的显示或隐藏是通过删除或添加ng-hide CSS 类到元素上来实现的。类似于设置 display 样式属性值。

ngShow为true，则显示；ngHide为true，则隐藏。可以同时设置两个标记，通过这两个指令，来根据属性的值，显示不同的内容。

3、ngChecked和ngDisabled：设置元素的 checked 属性或者 disabled 属性。

4、ngStyle 和 ngClass：设置元素的 style 和 class样式属性。

**案例：见test5\_使用其他指令.html。**

### 4.2.2. 【其他指令】事件指令

讲解事件指令。

ngClick指令：可以指定一个自定义的函数，实现单击事件的绑定。但是，此函数必须是在控制器中定义。传统的onXXX 都有一个对应的 ngXXX 版本。

**案例：见test5\_使用其他指令.html。此时，只讲解如何使用 ng-click即可，复杂操作放入第二天。**

# 5. 每日一练

**要求实现一个购物车，效果图如下：**