来源：https://segmentfault.com/a/1190000005836443

1. ng-if 跟 ng-show/hide 的区别有哪些？

第一点区别是，ng-if 在后面表达式为 true 的时候才创建这个 dom 节点，ng-show 是初始时就创建了，用 display:block 和 display:none 来控制显示和不显示。

第二点区别是，ng-if 会（隐式地）产生新作用域，ng-switch 、 ng-include 等会动态创建一块界面的也是如此。

这样会导致，在 ng-if 中用基本变量绑定 ng-model，并在外层 div 中把此 model 绑定给另一个显示区域，内层改变时，外层不会同步改变，因为此时已经是两个变量了。

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

<div ng-if="true">

<input type="text" ng-model="name">

</div>

ng-show 不存在此问题，因为它不自带一级作用域。

避免这类问题出现的办法是，始终将页面中的元素绑定到对象的属性（data.x）而不是直接绑定到基本变量（x）上。

1. ng-repeat迭代数组的时候，如果数组中有相同值，会有什么问题，如何解决？

会报错  
<li ng-repeat="item in items track by $index">{{ item }}</li>

"item in items track by $index"

1. ng-click 中写的表达式，能使用 JS 原生对象上的方法吗？

不能，DOM中不能使用原生js的方法。只能使用model里的方法，ng-click也一样。Model（scope）里面可以使用这些方法。

<p>{{parseInt(55.66)}}<p>

会发现，什么也没有显示。但如果在 $scope 中添加了这个函数：

$scope.parseInt = function(x){

return parseInt(x);

}

对于这种需求，使用一个 filter 或许是不错的选择：

<p>{{13.14 | parseIntFilter}}</p>

Angular自带过滤器，currency(转换为货币表示)，filter（从数组选取一个子集），lowercase，uppercase,orderBy(根据某个字符段属性重新排序)

还可以自己添加过滤器，方法如下：

app.filter('parseIntFilter', function(){

return function(item){

return parseInt(item);

}

})

以冒号的形式传参。

1. {{now | 'yyyy-MM-dd'}} 这种表达式里面，竖线和后面的参数通过什么方式可以自定义？

同上，过滤器，此时要传参。

// 形式

app.filter('过滤器名称',function(){

return function(需要过滤的对象,过滤器参数1,过滤器参数2,...){

//...做一些事情

return 处理后的对象;

}

});

// 栗子

app.filter('timesFilter', function(){

return function(item, times){

var result = '';

for(var i = 0; i < times; i++){

result += item;

}

return result;

}

})

1. angular 的数据绑定采用什么机制？详述原理

脏检测机制，model和view数据双向绑定。一次绑定对应一个switch监控，如果model有变化的东西，浏览器接收到事件就会遍历所有switch更新dom

双向数据绑定是 AngularJS 的核心机制之一。当 view 中有任何数据变化时，会更新到 model ，当 model 中数据有变化时，view 也会同步更新，显然，这需要一个监控。

原理就是，Angular 在 scope 模型上设置了一个 监听队列，用来监听数据变化并更新 view 。每次绑定一个东西到 view 上时 AngularJS 就会往 $watch 队列里插入一条 $watch，用来检测它监视的 model 里是否有变化的东西。当浏览器接收到可以被 angular context 处理的事件时，$digest 循环就会触发，遍历所有的 $watch，最后更新 dom。

举个栗子

<button ng-click="val=val+1">increase 1</button>

click 时会产生一次更新的操作（**至少触发两次 $digest 循环**）

* 按下按钮
* 浏览器接收到一个事件，进入到 angular context
* $digest 循环开始执行，查询每个 $watch 是否变化
* 由于监视 $scope.val 的 $watch 报告了变化，因此强制再执行一次 $digest 循环
* 新的 $digest 循环未检测到变化
* 浏览器拿回控制器，更新 $scope.val 新值对应的 dom

$digest 循环的上限是 10 次（超过 10次后抛出一个异常，防止无限循环）。

1. Factory service provider的区别

（补充）function test(){

this.a="hello";

return {

b:"hi",

c:"how are you"

}

}

var tes=new test();

alert(tes.a);

输出undifined。

Factory：返回一个对象

app.factory(“FooService”,function(){

return {

target:”factory”,

sayHello:function(){

return “hello”+this.factory}

}

}

Service：this方式返回一个对象。构造函数方式

app.service(“FooService”,function(){

var self=this;

this.target=”service”;

this.sayHello=function(){return “hello”+self.target}

})

Provider：创建一个可通过 config 配置的 service，$get 中返回的，就是用 factory 创建 service 的内容

app.provider('FooService', function(){

this.configData = 'init data';

this.setConfigData = function(data){

if(data){

this.configData = data;

}

}

this.$get = function(){

var self = this;

return {

target: 'provider',

sayHello: function(){

return self.configData + ' hello ' + this.target;

}

}

}

});

// 此处注入的是 FooService 的 provider

app.config(function(FooServiceProvider){

FooServiceProvider.setConfigData('config data');

});

从底层实现上来看，service 调用了 factory，返回其实例；factory 调用了 provider，返回其 $get 中定义的内容。factory 和 service 功能类似，只不过 **factory 是普通 function，可以返回任何东西**（return 的都可以被访问，所以那些私有变量怎么写，你懂的）；**service 是构造器，可以不返回**（绑定到 this 的都可以被访问）；provider 是加强版 factory，返回一个可配置的 factory。

https://www.oschina.net/translate/angularjs-factory-vs-service-vs-provider

controller 这一层应该很薄；也就是说，应用里大部分的业务逻辑和持久化数据都应该放在 service 里。"new"关键字实例化的

**使用Provider创建一个service的独特之处是，你可以在Provider对象传递到应用程序的其他部分之前在app.config函数对其进行修改。让我们来看一个对应的例子。**

1. 两个平级界面块 a 和 b，如果 a 中触发一个事件，有哪些方式能让 b 知道？详述原理

[**http://huangtengfei.com/2015/09/communication-in-controller/**](http://huangtengfei.com/2015/09/communication-in-controller/)

**Controller的通信，三种方式：（补充）**

1. **作用域继承。**利用子 Controller 控制父 Controller 上的数据。（父 Controller 中的数据要为引用类型，不能是基本类型，原因参见 [AngularJS中的作用域](http://huangtengfei.com/2015/09/scope-in-angularjs/) 一文）**【仅适用于上下级】**
2. 注入服务，把共享的数据注册为一个service，注入到controller。
3. 基于事件，利用angular的事件机制，使用 $on、$emit 和 $boardcast

Angular 为 $scope 提供了冒泡和隧道机制，$broadcast 会把事件广播给所有子 Controller，而 $emit 则会将事件冒泡传递给父 Controller，$on 则是 Angular 的事件监听函数，利用这三者，可以实现上下级和同级（需要构造一个共同的父级 Controller）之间的通信。

共用服务：factory生成一个对象，在a和b中注入这个对象。

基于对象：这个又分两种方式

第一种是借助父 controller。在子 controller 中向父 controller 触发（$emit）一个事件，然后在父 controller 中监听（$on）事件，再广播（$broadcast）给子 controller ，这样通过事件携带的参数，实现了数据经过父 controller，在同级 controller 之间传播。

第二种是借助 $rootScope。每个 Angular 应用默认有一个根作用域 $rootScope， 根作用域位于最顶层，从它往下挂着各级作用域。所以，如果子控制器直接使用 $rootScope 广播和接收事件，那么就可实现同级之间的通信。

1. 一个 angular 应用应当如何良好地分层？

**项目较小时，按文件类型**。Js/css/html/view/service/controller/fileter/templates

**项目较大时，按业务模块。**

##### 逻辑代码的拆分

作为一个 MVVM 框架，Angular 应用本身就应该按照 模型，视图模型（控制器），视图来划分。

这里逻辑代码的拆分，主要是指尽量让 controller 这一层很薄。提取共用的逻辑到 service 中 （比如后台数据的请求，数据的共享和缓存，基于事件的模块间通信等），提取共用的界面操作到 directive 中（比如将日期选择、分页等封装成组件等），提取共用的格式化操作到 filter 中等等。

在复杂的应用中，也可以为实体建立对应的构造函数，比如硬盘（Disk）模块，可能有列表、新建、详情这样几个视图，并分别对应的有 controller，那么可以建一个 Disk 构造函数，里面完成数据的增删改查和验证操作，有跟 Disk 相关的 controller，就注入 Disk 构造器并生成一个实例，这个实例就具备了增删改查和验证方法。这样既层次分明，又实现了复用（让 controller 层更薄了）。

1. angular 的缺点有哪些？

强约束，对前端不友好。

不利于SEO，内容都是动态生成的。（可解决）

一种解决办法是，对于正常用户的访问，服务器响应 AngularJS 应用的内容；对于搜索引擎的访问，则响应专门针对 SEO 的HTML页面。

作为 MVVM 框架，因为实现了数据的双向绑定，对于大数组、复杂对象会存在性能问题。

可以用来 [优化 Angular 应用的性能](https://github.com/xufei/blog/issues/23) 的办法：

1. 减少监控项（比如对不会变化的数据采用单向绑定）
2. 主动设置索引（指定 track by，简单类型默认用自身当索引，对象默认使用 $$hashKey，比如改为 track by item.id）
3. 降低渲染数据量（比如分页，或者每次取一小部分数据，根据需要再取）
4. 数据扁平化（比如对于树状结构，使用扁平化结构，构建一个 map 和树状数据，对树操作时，由于跟扁平数据同一引用，树状数据变更会同步到原始的扁平数据）
5. 如何看待angular2

组件化，动态加载module

1. 详述 angular 的 “依赖注入”

依赖注入是一种软件设计模式，目的是处理代码之间的依赖关系，减少组件间的耦合。

**DI（依赖注入）容器**的三个要素：服务的注册，依赖的声明，对象的获取。

在angular中，module和$provide相当于是服务的注册；injector用来获取对象（angular会自动完成依赖的注入）；依赖关系的声明在angular中有3种方式。下面从这3个方面，介绍下angular的DI。

angular.module(name, [requires], [configFn]);

name：字符串类型，代表模块的名称；

requires：字符串的数组，代表该模块依赖的其他模块列表，如果不依赖其他模块，用空数组即可；

configFn：用来对该模块进行一些配置。

angularjs中与DI相关有angular.module()、angular.injector()、 [$injector](http://docs.angularjs.cn/api/auto/service/$injector)、[$provide](http://docs.angularjs.cn/api/auto/service/$provide)。

可以看到$provide提供了provide()、constant()、value()、factory()、service()来创建各种不同性质的服务；angular.Module中也提供了这5个服务注册方法。其实2者功能是完全一样的，就是用来向DI容器注册服务到injector中。