### 静态页面和网站

1. 静态页面：页面就是一个完整的html；

动态页面是一个html，页面的内容由js生成。

1. 静态网站和动态网站：服务器的能力。静态网站服务器不做任何事情，只是简单地将请求页面发送给浏览器。一般来说，网址上以.html结尾的都是静态网站。动态网站是指服务器接收用户的请求和参数，根据参数在服务器端执行相应的代码（php, c#, java...），**拼接字符串**合成html文本，并将其返回给浏览器。浏览器拿到的是一个已经处理了的html页面，它不需要做任何事情，只需要直接显示即可。
2. 库library

库就是存储已经封闭好的js对象、方法、函数、常量等内容的地方。

1. 框架framework

提供了一套规则与执行方案，我们只需要按照规则编写代码，不用考虑内容如何执行。

### Angular概述

1. AngularJS是一个JavaScript框架，致力于单页面应用程序SPA: single page application
2. AngularJS易于构建页面CRUD(Create / Retrieve / Update / Delete)操作，包括：数据绑定、基本模板标识符、表单验证、路由、深度链接、组件重用、依赖注入。
3. 基本操作思路与DOM与jQuery的先查找元素再操作元素不同，它是一切操作都以数据为中心（创建、绑定、修改、更新数据）。

### Angular四大特性

1. 采用MVC架构模型
2. 双向数据绑定
3. 依赖注入
4. 模块化设计

### Angular指令

1. 指令：angular中以ng开头的属性叫指令

ng-app告诉angular来管理html代码，管理ng-app所在元素及子元素。

ng-click 注册点击事件（onclick），后面“”的代码相当于匿名函数。

选择器 <input type = “text” ng-model = “user.name”>

<body ng-app ng-init=“user.name = ‘Lily’”> 初始化

1. 表达式expression

JS表达式：将运算符与数据（变量）连接起来，可以计算得到结果的式子（序列）就是表达式。例如：123是常量表达式，“1+2+3;”不是表达式，是语句（计算出来有结果，也就是说可以赋值给另一个变量，也可以用括号将其括起来）。

ng表达式：是一个类似于js代码的式子，用在ng-xxx指令中，或{{}}这类语法中。

**A**ngular Expressions are JavaScript-like code snippets that are mainly placed in interpolation bindings such as as <span title="{{ attrBinding }}">{{ textBinding }}</span>, but also used directly in directive attributes such as ng-click="functionExpression()".

For example, these are valid expressions in AngularJS:

* 1+2
* 3\*10 | currency
* user.name
* items[index]

　表达式（Expressions）是类Javascript的代码片段，通常放置在绑定区域中（如｛｛expression｝｝）。表达式通过$parse服务（http://code.angularjs.org/1.0.2/docs/api/ng.$parse）解析执行。

### Angular表达式 vs. Js 表达式

　　这很容易让人将angular视图表达式联想为javascript表达式，但这并不完全正确，因为angular不是通过javascript的eval()对表达式进行求值。你可以将angular表达式想象为带有以下差异的javascript表达式：

* 属性求值：所有属性的求值是对于scope的，而javascript是对于window对象的。
* 宽容（forgiving）：表达式求值，对于undefined和null，angular是宽容的，但Javascript会产生NullPointerExceptions。
* 没有流程控制语句：在angular表达式里，我们不能做以下任何的事：条件分支、循环、抛出异常。
* 过滤器（filters）：我们可以就将表达式的结果传入过滤器链（filter chains）。例如将日期对象转换为本地指定的人类可读的格式。

　　另一方面，如果我们想（在angular表达式中）执行任意的Javascript代码，我们可以将那些代码写到Controller的一个方法中并调用它。如果我们想在javascript中eval()一个angular表达式，可以使用$eval()方法。

1. 上下文
2. 没有函数声明
3. 没有正则字面量
4. 不能使用new关键字来实例化对象
5. 不能使用位运算、逗号运算、void运算。

<input type="text" ng-model = "(val1-0)+(val2-0)">

<input type="text" value = "{{(val1-0)+(val2-0)}}">

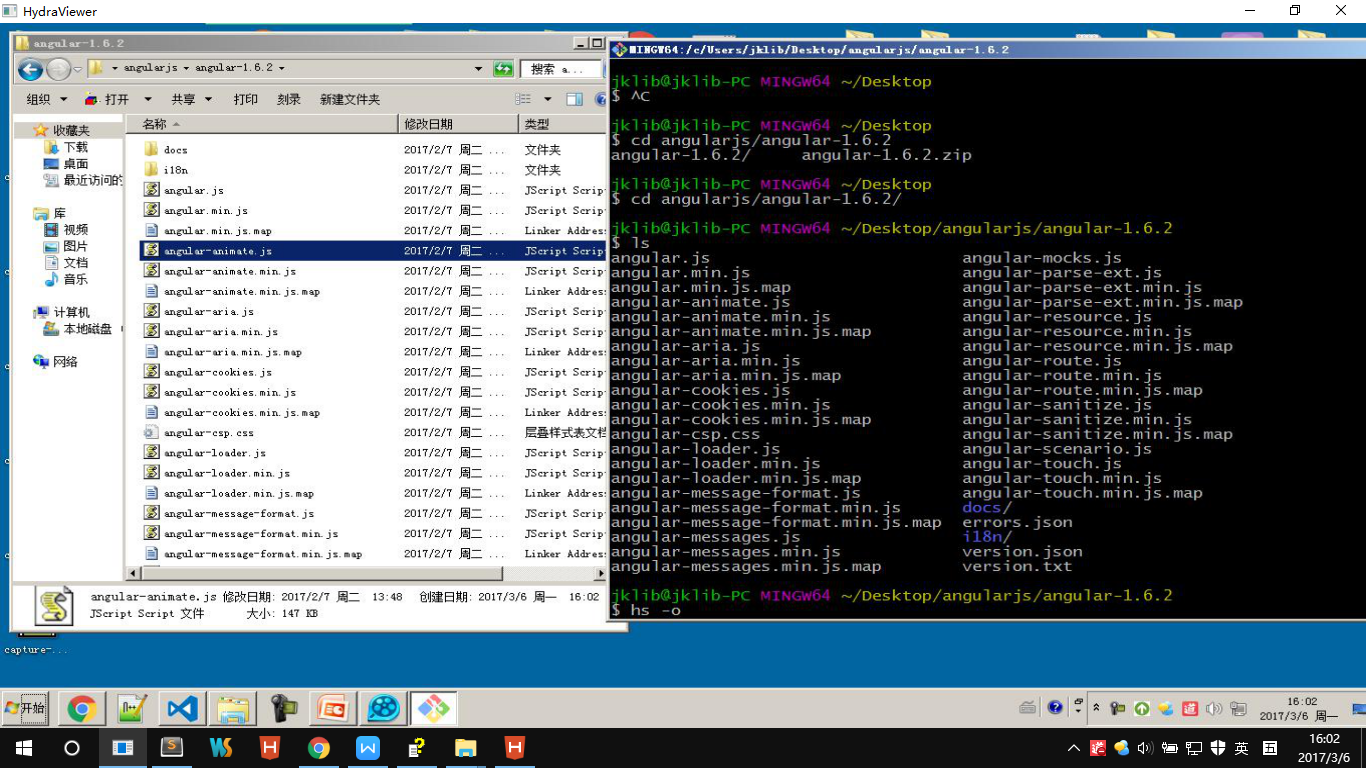
http-server

安装：npm install -g http-server

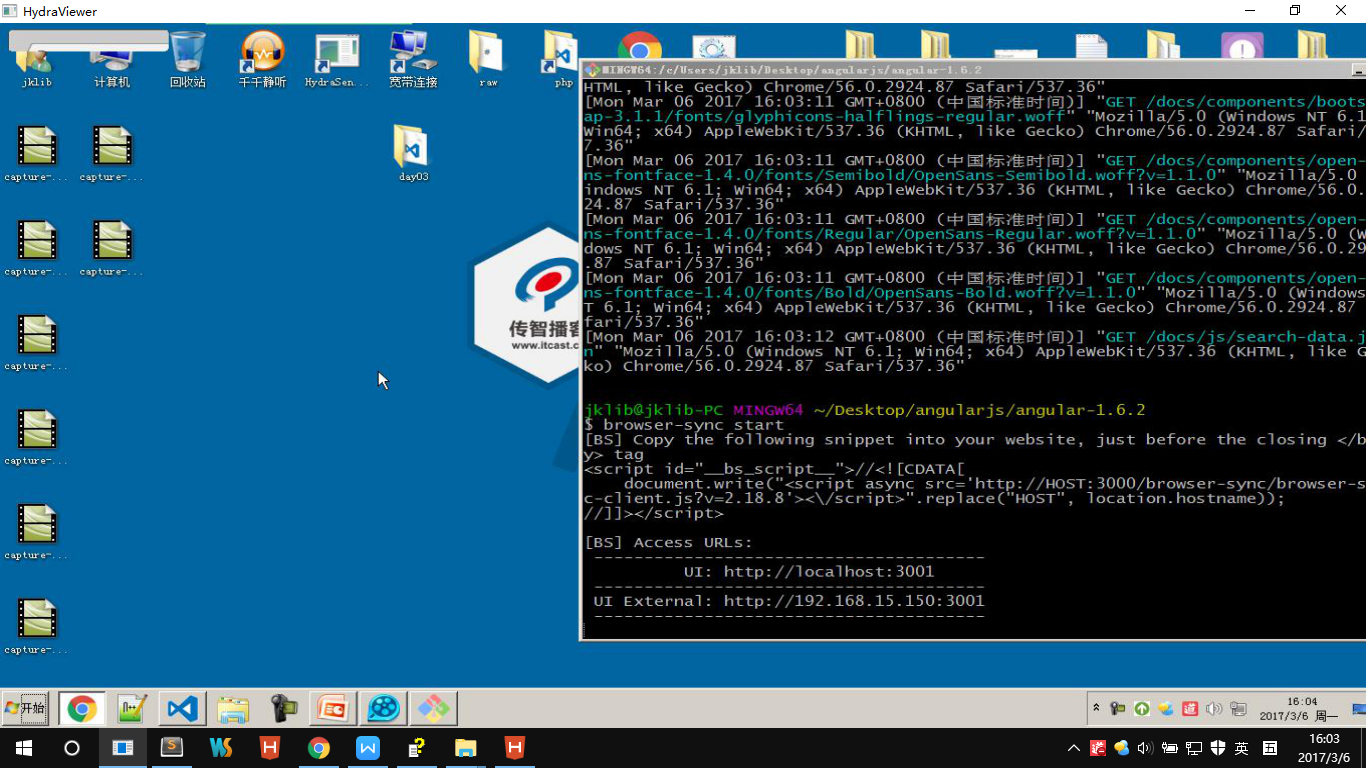
查看版本：$http-server --version

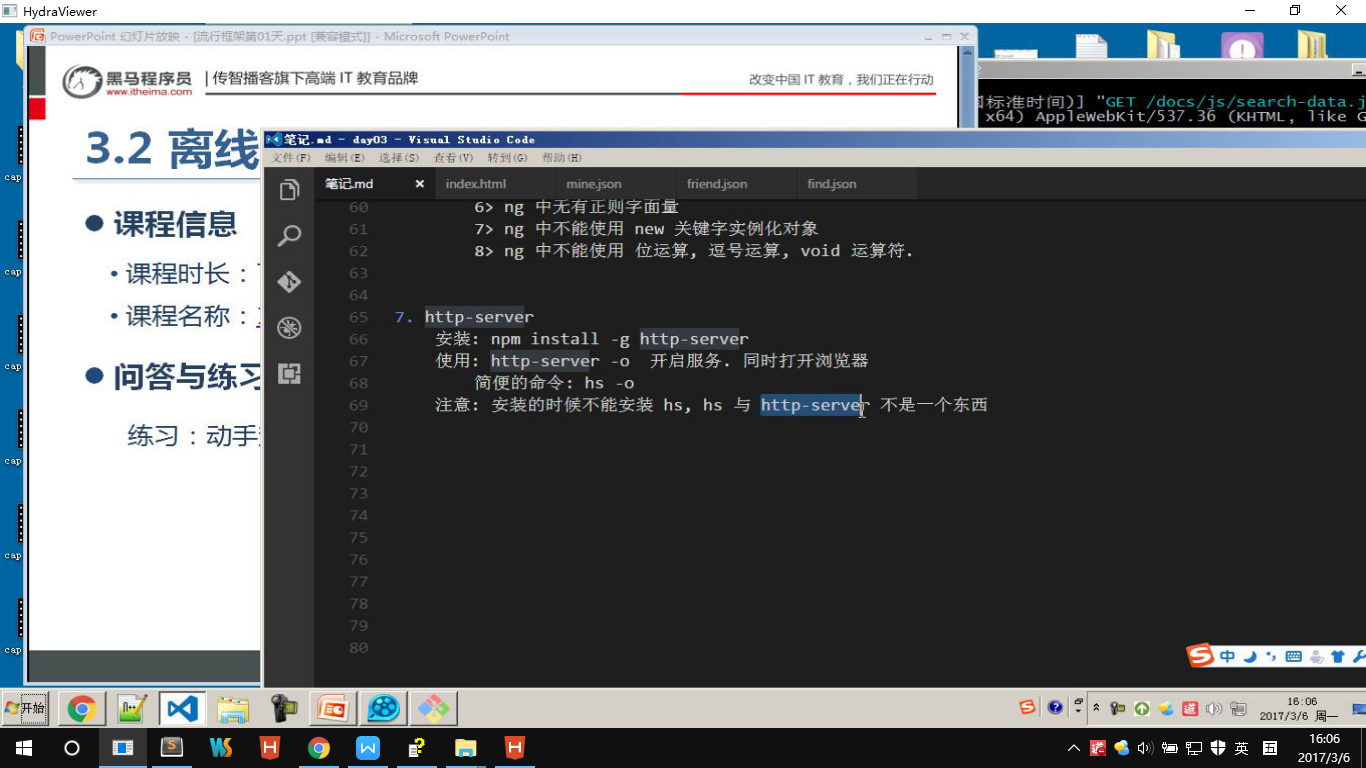
然后把网站根目录放在 angular-docs 下

简写hs-o即可。



也可以用browser-sync:

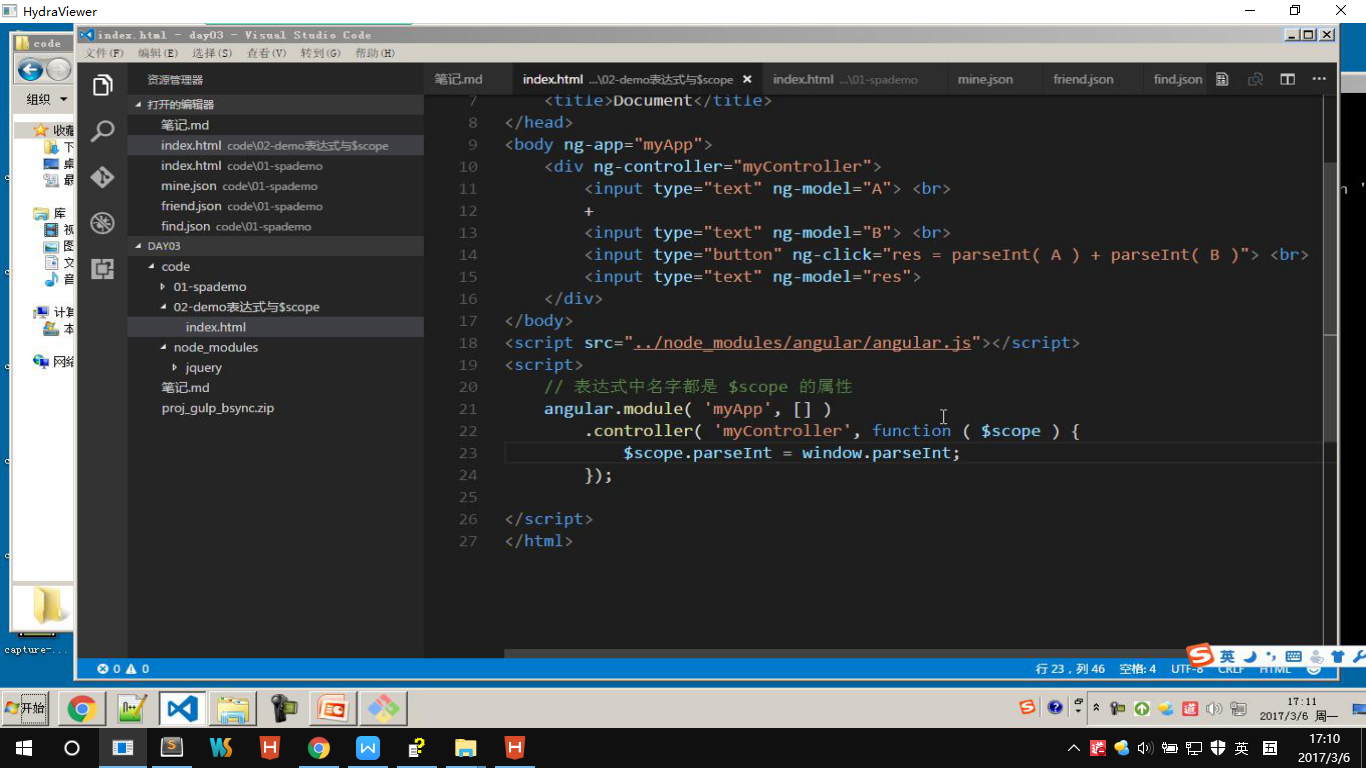




Angular可以创建模块，也可以用变量接收该模块，还可以获取该模块，为什么要有这种方法？

直接获取模块就不需要用变量接收，这样可以减少全局污染；创建以后返回的是该模块是为了方便实现链式编程。

表达式中的名字都是$scope的属性。



MVC

M model 模型，数据

V view

C controller

### Understanding Controllers

In AngularJS, a Controller is defined by a JavaScript **constructor function** that is used to augment the [AngularJS Scope](https://docs.angularjs.org/guide/scope).

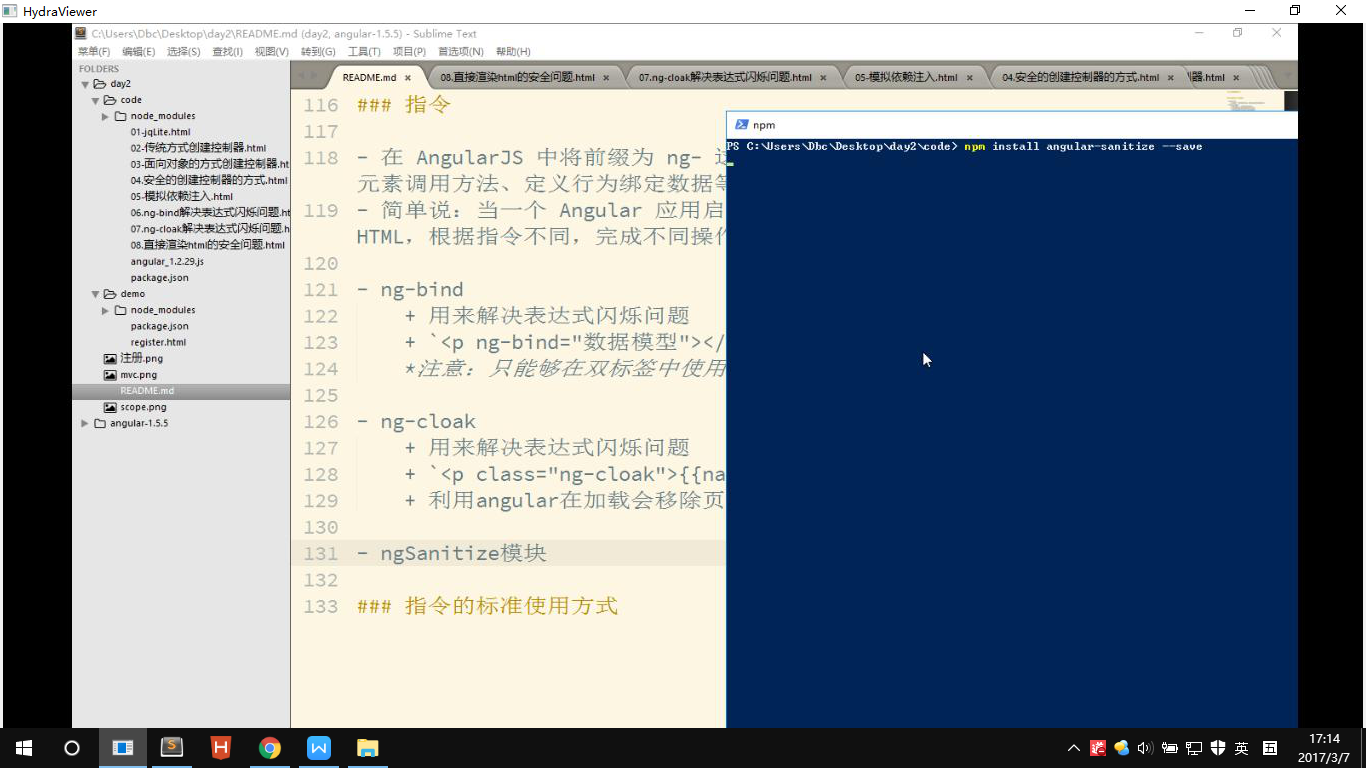
When a Controller is attached to the DOM via the [ng-controller](https://docs.angularjs.org/api/ng/directive/ngController) directive, AngularJS will instantiate a new Controller object, using the specified Controller's **constructor function**. A new **child scope** will be created and made available as an injectable parameter to the Controller's constructor function as $scope.

If the controller has been attached using the controller as syntax then the controller instance will be assigned to a property on the new scope.

Use controllers to:

* Set up the initial state of the $scope object. 初始化
* Add behavior to the $scope object. 划分$scope对象

npm init -f



依赖注入Dependency injection，简单地说就是参数传递。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **指令** | | | | | | |
| ng-app | | | | 相当于一个入口，告诉angular来管理页面的html代码 | | |
| ng-click | | | | 用于注册点击事件 | | |
| ng-model | | | | 用于进行**双向**数据绑定 | | |
| ng-init | | | | 用来初始化数据模型  <body ng-app ng-init = "name ='Lily'"> | | |
| ng-module | | | | 创建或获取模块angular.module("mainApp",[]) | | |
| ng-controller | | | | 指向了创建的控制器  angular.module("myApp",[]).controller(“myCtrl”,function($scope){}); | | |
| ng-bind  解决闪烁问题 | | | | 能够绑定数据模型的值，只能作用于双标签元素  <p ng-bind = "name"></p>  $scope.name = "Lily"; | | |
| ng-cloak | | | | angular会移除所有样式名为ng-cloak的样式的特性  .ng-cloak { display: none; }  <p class = "ng-cloak">{{name}}</p> | | |
| ng-bind-html  引入sanitise模块 | | | | 用于安全地渲染html代码  <p class = "ng-bind-html">{{name}}</p> | | |
| ng-repeat  循环输出数组 | | | | <li ng-repeat = “item in data”></li>  遍历对象的话，item对应的是每个key的value  <li ng-repeat = “item in data track by $index”></li>  按照索引来循环避免重复 | | |
| ng-class | | | | <li ng-class = '{"a":"red", "b":"blue", "c":" green"}[item.salary]'></li>从多个样式中选取一个  {“样式名001”：布尔值，“样式名002”：布尔值}  class = “{{三元表达式 }}” | | |
| ng-hide/ng-show | | | | <li ng-hide = "item.salary == 'a'"></li>  显示或隐藏页面元素 | | |
| ng-if | | | | 布尔值为true时显示，false时删除 | | |
| ng-switch  ng-switch-when | | | | <div ng-switch = "name">  <div ng-switch-when = "andy">I am Andy.</div>  </div> | | |
| ng-submit | | | |  | | |
| **功能性指令** | | | | | | |
| ng-checked | | | ng-selected | | ng-disabled | ng-readonly |
| 只能进行**单向**数据绑定，从控制器操作才有效 | | | | | | |
| **事件指令** | | | | | | |
| ng-blur | |  | | | | |
| ng-focus | |  | | | | |
| ng-click | |  | | | | |
| ng-dbclick | |  | | | | |
| ng-change | | <input ng-change="fn()"></input> <select><textarea>  输入框的值改变时执行函数,只针对输入框值的真实修改 | | | | |
| ng-copy | | <input ng-copy="fn()"></input>  输入框的文本被拷贝时执行表达式 | | | | |
| ng-submit | | <form ng-submit="func()"></form>  用于在表单提交后执行指定函数 | | | | |
| **数据模型** | | | | | | |
| $even | | | $first | | $middle | $log |
| $odd | | | $last | |  | $location |
| **方法** | | | | | | |
| $watch | $scope.$watch("name",function(nowValue,oldValue){}) | | | | | |
| $filter | $filter("date")($scope.date,"yyyy-mm-dd")  第一个参数是过滤器的名称，第二个是要过滤的数据，第三个是其他参数 | | | | | |

limitTo 多用于限制字符串长度

$scope.location = $location; $scope.$watch("location.url()",function(nowValue,oldValue){

},true) //true表示是否深度监听每一个参数的每一个属性的变化

Q：为什么不能直接监视"$location.url()"？

A：$watch**只**能用于监视数据模型的变化，无法监视$scope / $location等注入的模型。$scope.$watch里的回调函数会默认执行一次。

Q：<input type="checkbox" ng-model= "toggle"> 和<input type="checkbox" ng-checked = "toggle"> 有什么不同？

A：前者随着点击，$scope.toggle 会在true和false之间切换，可以打印出来看看。

### 过滤器

|  |  |
| --- | --- |
| date | 将从1970年开始的毫秒数转成日期的形式 |
| currency | 将数字转成货币的形式，可通过冒号传递参数  <p>{{money|currency:'￥'}}</p> |
| filter | <li ng-repeat = "item in tasks | filter: false"></li>  与ng-repeat配合使用，普通参数就全局匹配，对象就匹配属性 |
| json |  |
| limitTo | 根据传入的参数生成一个新的数组或字符串  <p>{{string|limitTo:5:2}}</p> //从index=2开始截取5个  <p>{{string|limitTo:-5}}</p> //从后往前截取5个 |
| lowercase |  |
| uppercase |  |
| orderBy | 按某个顺序排列  <p ng-repeat = "item in arr | orderBy:'age'">  <p ng-repeat = "item in arr | orderBy:'-age'"> 逆序 |
| number |  |

### 自定义指令

1. 通过模块对象的directive方法创建，有两个参数，第一个是指令的名字，必须使用驼峰命名法，但使用时要用小写并用-分割；第二个参数和控制器的第二个参数一样，在其中的function里返回一个obj对象。
2. 返回的对象的属性：

|  |  |
| --- | --- |
| **KEY** | **VALUE** |
| template: | 需要一个字符串，最终这个字符串将被添加到自定义指令所在的标签的innerHTML位置 |
| templateUrl: | 需要一个字符串，可以是一个html文件的路径，angular最终会异步请求这个文件，把文件内容插入到自定义指令所在的标签的innerHTML位置；  也可以是一个script标签的id地址，把script标签中的内容插入到自定义指令所在的标签的innerHTML位置 |
| <script id = "tem" type = "text/ng-template">  <button>I am a button.</button>  </script>  templateUrl: "tem" |
| restrict: | 限制自定义标签的使用方式：  “ECMA” element,class,comment,attribute |
| replace: | 为true时替换自定义指令所在的标签，模板中只能有一个标签。  不能与transclude同时使用 |
| transclude: | 需要与ng-transclude配合使用。为true时  1. 把自定义指令所在标签的innerHTML放在 <span></span> 中  2. 然后把<span></span>放到模板中由ng-transclude指定的元素的innerHTML位置（该元素原来的内容被替换）  3. 再把模板放到自定义指令所在的标签的innerHTML位置 |
| scope: | {myattr: “@attr”} @后面的值是自定义指令所在标签的属性名，最终可以在模板字符串中使用表达式来获取相应的属性值，可简写为{test: “@”} true / false / {}  作用是让内容带着某种属性一起走。  不能与link同时使用 |
| link: | function (scope, element, attributes){  element.on(“click”, function(){  console.log(“this”);  })  }  scope类似控制器中的 $scope，可以暴露一些值  element 是自定义指令所在标签的jqLite对象  attributes是自定义指令所在标签的所有属性的集合 |
| t t r r t s l 天天嚷嚷特斯拉 | |

### Todos项目

1. 往localStorage中存储数据时，数组中的每个对象的id属性如何产生唯一的id值？
2. 用Math.random()会有极小概率的重复
3. sha1和 md5都可以产生唯一号
4. 用new Date() 时间戳
5. 用数组最后一个元素（即一个对象）的id值。
6. localStorage.getItem(“ \* ”) 取到的是json形式的字符串，如果localStorage中没有相应的数据，那么取到的是**“[]”**.
7. this指向问题： todos的add和remove方法中都调用了this.save方法。

$scope.add = function(){

if(!$scope.newTask){

return;

}

todos.add($scope.newTask);

//add由todos调用，所以this指向todos

}

$scope.remove = todos.remove;

//remove由$scope调用，所以remove中调用的save中的this指向的还是$scope

this.save = function(){

$window.localStorage.setItem("todos",JSON.stringify(tasks));

}

1. track by $index要放到过滤器后面写

<li ng-repeat = "item in tasks|filter:finish track by $index"></li>

1. controller 要想取到service里的数据，有两种方法：
2. 将service里的数据赋给this的某个属性，实例对象就可以把这些数据带到controller里面，再赋值给$scope的某个属性。

this.tasks = tasks;

$scope.tasks = todos.tasks;

1. 用函数将数据返回，实例对象调用这个函数就可以得到这些数据。

this.get = function(){

return tasks;

}

$scope.tasks = Todos.get();

### 路由

1. location是BOM对象的一个属性，它也是一个对象，它有若干属性

|  |  |
| --- | --- |
| 属性名 | 属性值 |
| location.hash | "#/active" |
| $location.url() | /active 也是字符串，但是没有# |
| location.href | "file:///D:/wamp/www/test/todos/index.html#/" |
| location.search | "?para=5806" 用于获取参数 |

<body ng-app = "myApp">

<div>1</div>

<div ng-view>2</div> //模板内容要显示的位置

<script src="../lib/angular-1.5.5/angular.js"></script>

<script src="../lib/angular-1.5.5/angular-route.js"></script>

<script>

angular.module("myApp",["ngRoute"]).config(["$routeProvider",function($routeProvider){

$routeProvider.when("/students/:anchor?",{

//when用于监测锚点的变化

template: "<p>{{student.id}}{{student.name}}</p>",

//也可以用templateUrl

controller: "myCtrl" //控制器暴露的数据可以在模板中使用

}).otherwise({

redirectTo: "/students/"

})

}]).controller("myCtrl",["$scope","$routeParams","$route",function($scope,$routeParams,$route){

$scope.student = {

a:{id:1,name:"Andy",age: 2},

b:{id:1,name:"Bob" ,age: 4},

c:{id:1,name:"Ciro",age: 6},

d:{id:1,name:"Deny",age: 8}

}

$scope.student = $scope.student[$routeParams.anchor];

//$routeParams是一个拥有anchor属性的对象，anchor?表示参数是可控的，**/**后面有没有内容都可以执行匹配的动作。

if(!$scope.student){

$route.updateParams({anchor: "d"});

}

}])</script>

$routeParams对应的是锚点值中冒号后面的一部分，所以$route.updateParams改变的只是锚点的一部分。如果要改变整个锚点值，需要使用$location.url()。$location 是全路由，是对全部路由进行变更。

$route是在页面加载完成以后才起作用，在同一个路由下变更锚点。

**服务**

服务名和$scope / $location都要作为控制器第二个参数（一个函数）的参数传入，此时服务名代表由服务的第二个参数（一个构造函数）new出来的**实例对象**。