AngularJS--第二天 课程

模块化

开发时候

```
m1.controller('Aaa',function($scope){
    $scope.name='hello';
});
```

上线时候，代码需要进行压缩

function($scope)经过压缩会变成function($s) angular不会识别$s

写成一个数组方式，可以解决这个问题

```
m1.controller('Aaa',['$scope',function($scope){
    $scope.name='hello';
}]);
```

angularJS的工具方法

angular.bind();

angular.extend();

angularJS提供了很多工具方法

angular.bind(); -->改this指向

```
function show(n1,n2){
    alert(n1);
    alert(n2);
    alert(this);
}
angular.bind(document,show)(3,4);
```

angular.copy(); -->拷贝对象

```
var a = {
  name :  'hello'
};
var b = {
  age :  '20'
};
var c = angular.copy(a,b);   //a把所有值覆盖给了b
console.log(b);
```

angular.extend(); -->对象继承

```
var a = {
  name :  'hello'
```

```
};
var b = {
  age :  '20'
};
var c = angular.extend(b,a);   //a把所有值覆盖给了b
console.log(b);
```

angular工具方法

angular.isArray　　//判断是否是数组
angular.isDate　　//判断是否是时间
angular.isDefined　　//判断元素是否是存在
angular.isUndefined　　//判断元素是否是不存在
angular.isFunction　　//判断是否是方法
angular.isNumber　　//判断是否是数字
angular.isObject　　//判断是否是对象
angular.isString　　//判断是否是字符串
angular.isElement　　//判断是否是元素

```
 window.onload=function(){
            console.log(angular.isElement(document.body));
        }

 // console.log(angular.isElement($(document.body) ));
```

也支持jq对象

angular.version　　//判断当前angular的版本

```
console.log(angular.version);
```

angular.equals　　//判断两个元素是否相等

```
var a = 1;
var b = 1;
console.log(angular.equals(a,b));

 // 两个空数组  []是true   两个NAN也是true
 // js中这两个例子都是不等的
 // angular.equals(); 会判断两个相同的就是真，不同的就是假。
```

angular.forEach　　//遍历

```
var values = ['a','b','c'];
var values = ['name','hello','age','20'];
angular.forEach(values,function(valur,i){
      console.log(value);
      console.log(i);
      this.push(value + i);
}resule);
console.log(result);
```

angular.fromJson/toJson　//对字符串格式的json解析和对json字符串的转换

```
var str = ['name','hello','age','20'];
var json = angular.fromJson(str);
console.log(json);

var json = ['name','hello','age','20'];
var str = angular.toJson(json);
console.log(str);
```

angular.identity/noop　第一个是返回传递的变量　noop是一个undefined

```
var str=  'hello';
console.log(angular.identity(str));   //结果是hello

function  identity(str){
    return str;          //和这个是一样的
}

console.log(angular.noop());   //undefined
function noop(){

}
```

辅助作用~在一些编程过程中~需要一些默认的设置

angular.lowercase/uppercase 转换大写小写

```
console.log(angular.upparcase('hello'));把小写转换大写
```

angular.element.　可以使用jQ中的部分方法

```
angular.element(oDiv).css('background','red')
```

angular.bootstrap　初始化作用、动态添加ng-App

```
document.onclick = function(){
    angular.bootstrap(document.,['myApp'])

}
```

当有多个初始化的时候。只能识别一个！
可以动态添加多个初始化

angular.injector　注册器（内部工具方法）

$scope
$scope.$watch
监听

$scope.$apply

放入原生js中，让原生代码可以实现功能

```
<!DOCTYPE html>
<html ng-app="myApp">
```

```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',function($scope,$timeout){
        $scope.name='hi';
        setTimeout(function(){
            $scope.$apply(function(){
                $scope.name='你好';
            });
        // $scope.$apply可以监听 $scope.name='你好';当数值有变化可以改变视图
        // 可以实现原生和第三方
                },2000);
//  setTime方法
                //  要用angular提供的服务$timeout
                // $timeout(function(){
                //     $scope.name='你好';
                // },2000);
    })
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
        <p>{{ name }}</p>
    </div>
</body>
</html>
```

$scope.$apply这个方法$timeout、和ng-click之类的都会内部默认调用的

angular.module

controller

控制器

run()

可以初始化全局数据$rootScope

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    // m1.controller('Aaa',function($scope,$timeout){
    //     $scope.name='hi';

    // })
    m1.run(['$rootScope',function($rootScope){
```

```
        $rootScope.name='hello';
    }])
    </script>
</head>
<body>
    <div>
        <p>{{ name }}</p>
    </div>
</body>
</html>
```

run()没有$socpe局部作用域
可以不用控制器。直接刷新页面

过滤器

Currency:可以把数字转换金额形式

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',["$scope",function($scope){
        $scope.name='2352364326';
        // currency:可以把数字转换金额形式，默认是$可以改成￥
        // 用类似传参的方式实现{{ name | currency:'￥'}}
    }])
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
        <p>{{ name | currency:'￥'}}</p>
    </div>
</body>
</html>
```

{{数据 | currency : ￥}}

number：可以把数字转换千分符形式

```
// number:可以把数字转换千分符形式，默认是保留小数点后三位可以通过传参改成{{ name | number : 2}} :0是不保留
<p>{{ name | number : 2}}</p>
```

lowercase/uppercase

```
  // lowercase/uppercase：转换数据的大小写{{ name | uppercase }}小写转换大写  {{ name | lowercase }}大写转换小写
```

json：转换成json格式：转换成json的格式

```
// $scope.name={'name':'boy','age':'20'};
 <!-- <pre>{{ name | json }}</pre> -->
```

一定要用<pre>标签：可以识别空格和换行。

limitTo：截取

```
<p>{{ name | limitTo :  2}}</p>
// limitTo：可以支持字符串和数组截取几位根据参数实现<p>{{ name | limitTo :  2}}</p>
```

date:时间的格式

```
      <p>{{ name | date : 'yyyy'}}
 // date:把一组数字转换成时间的格式  也有很多参数  fullDate  yyyy  longDate  很多
```

orderBy：数组的排序

```
$scope.name = [
          {color:'red',age:'20'},
          {color:'blue',age:'30'},
          {color:'green',age:'40'},
          {color:'gray',age:'10'}
          ];
 //orderBy:数组的排序,数组必须要有一定的格式。可以根据相关参数排序。当第二参数为: true的时候可以实现逆向排序
<p>{{ name | orderBy : 'age' : true}}</p>
```

filter：

```
<p>{{name | currency}}</p>
```

变成金额 默认是$

```
<p>{{name | currency  : '￥'}}</p>
```

变成金额 ￥

```
<p>{{name | number}}</p>
```

变成数字 默认是小数点后三位

```
<p>{{name | number : 2}}</p>
```

变成小数点后两位

```
<p>{{name | uppercase}}</p>
```

转换成大写

```
<pre></pre>这个标签可以识别空格和换行
<pre>{{name | json}}</pre>
```

转换成json的格式

```
<p>{{name | limitTo : 2}}</p>
```

可以支持字符串和数组可以截取几位目前是2

```
<p>{{name | date}}</p>
```

时间的格式 也有很多参数 fullDate yyyy longDate 很多

数组的排序

```
<p>{{name | orderBy}}</p>
```

数组必须要有一定的格式

```
$scope.name = [
          {color:'red',age:'20'},
```

```
        {color:'blue',age:'30'},
        {color:'green',age:'40'},
        {color:'gray',age:'10'}
        ];
```

有参数

```
<p>{{name | orderBy  : color}}</p>
```

按照color的从小到大排列

按照age的从小到大排列

```
<p>{{ name | orderBy : 'age' : true}}</p>
```

当第二参数为: true的时候可以实现逆向排序

搜索

```
<p>{{name | filter  : '要搜索的值'}}</p>
```

可以模糊搜索

```
<p>{{name | filter  : '要搜索的值'  : 。true}}</p>
```

也可以精确搜索

过滤器的组合使用

可以通过管道符号 | ，分隔多个过滤器进行组合使用

```
<p>{{name | limitTo : 2  | uppercase }}</p>
```

先截取字符串2位 在转成大写

用$filter实现在js中使用过滤器

```
$scope.name = $filter('uppercase')('hello');

$scope.name = $filter('number').('1231244214.214124',1)

$scope.name = $filter('date')('124124',MMMM);
```