

## AngularJS--第一天 课程

### AngularJS简介--版本

<http://www.angularjs.org/>

AngularJS官网需要翻墙才能访问

下面是AngularJS社区有相关的资源下载

<http://www.github.com/angular/>

<http://www.angularjs.cn/>

<http://www.ngnice.com/>

<http://www.sinaapp.com/>

node.js

AngularJS下载

<http://www.bootcdn.cn/angular.js/>

npm install angular



**Misko Hevery** & Adam Abrons  
Google Since 2009

诞生于2009年，后来被Google收购

AngularJS 是什么？

AngularJS的官方文档是这样介绍它的。

完全使用JavaScript编写的客户端技术。同其他历史悠久的Web技术（HTML、CSS和JavaScript）配合使用，使Web应用开发比以往更简单、更快捷。

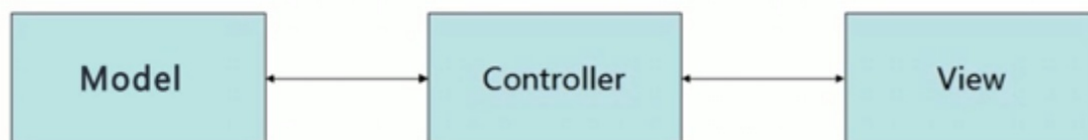
AngularJS主要用于构建单页面Web应用。它通过增加开发人员和常见Web应用开发任务之间的抽象级别，使构建交互式的现代Web应用变得更加简单。

简单理解就是，使我们更加方便快捷的开发web页面。

## Angular---4大核心特性

- 1.MVC
- 2.模块化
- 3.指令系统
- 4.双向数据绑定

### AngularJS核心特性1---MVC



**起源：**1979年，Trygve Reenskaug第一次正式提出了MVC模式

**Model：**数据模型层

**View：**视图层，负责展示

**Controller：**业务逻辑和控制逻辑

**好处：**职责清楚，代码模块化

### 第一个代码

```
<!DOCTYPE html>
<html ng-app>
//ng-app初始化的指令。如果没有ng-app指令angular是不会执行的
<head>
  <meta charset="utf-8">
  <script src="angular.min.js"></script>
</head>
<body>
  <input ng-model="name" type="text" placeholder="Your name">
  //ng-model是数据 绑定了一个值name
  <h1>Hello {{ name }}</h1>
</body>
</html>
```

虽然这个例子不怎么有趣，但它展示了AngularJS最基本也最令人印象深刻的功能之一：数据绑定。

`{{ name }}`是双花括号（表达式）作用是绑定！

## 模块化

模块化开发思想：1、减少全局的污染、和命名的问题。

2、可以做到模块之间的互相依赖，不用手动的进行依赖的处理。

如何做到模块化：

```
angular.module("myApp",[]);
```

有两个参数 1.一个是模块的名字myApp

2.一个是依赖模块的数组 [] （如果目前不需要其他的模块，那就一个空数组就行）

```
var m1 = angular.module("myApp",[]);
```

相应的初始化也需要绑定模块名：ng-app="myApp"

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

## 作用域

### \$Scope: 局部作用域

**ng-controller: 是控制器 是有作用域的**

案例一

```
<!DOCTYPE html>
<html ng-app="myApp">
  <head>
    <meta charset="{CHARSET}">
    <title></title>
    <script src="js/angular.min.js"></script>
    <script>
      var m1 = angular.module('myApp',[]);
      m1.controller('Aaa',function($scope){
        $scope.name='hello';
        $scope.age='20';
      });
    </script>
  </head>
  <body>
    <div ng-controller="Aaa">
      <p>{{name}}</p>
      <!-- <p>{{age}}</p> -->
    </div>
    <p>{{age}}</p>
  </body>
</html>
```

案例二

```
<!DOCTYPE html>
<html ng-app="myApp">
  <head>
    <meta charset="{CHARSET}">
    <title>作用域</title>
```

```

<script src="js/angular.min.js"></script>
<script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',function($scope){
        $scope.name='hello';
    });
    m1.controller('Bbb',function($scope){
        $scope.name='hi';
    });
</script>
</head>
<body>
    <div ng-controller="Aaa">
        <p>{{name}}</p>
    </div>
    <div ng-controller="Bbb">
        <p>{{name}}</p>
    </div>
</body>
</html>

```

## \$rootScope: Angular中的全局作用域

### 案例三：全局和局部

```

<!DOCTYPE html>
<html ng-app="myApp">
    <head>
        <meta charset="utf-8">
        <title>作用域</title>
        <script src="js/angular.min.js"></script>
        <script>
            var m1 = angular.module('myApp',[]);
            m1.controller('Aaa',function($scope,$rootScope){
                $scope.name='hello';
                $rootScope.age='20';
                // 当没有局部变量后，才会查找全局变量
            });
            m1.controller('Bbb',function($scope){
                $scope.name='hi';
                $scope.age='30';
                // 局部变量优先查找，类似js中的变量的作用域
            });
        </script>
    </head>
    <body>
        <div ng-controller="Aaa">
            <p>{{name}}</p>
        </div>
        <div ng-controller="Bbb">
            <p>{{name}}</p>
            <p>{{age}}</p>
        </div>
    </body>

```

</html>

#### 案例四：父子关系中的作用域

```
<!DOCTYPE html>
<html ng-app="myApp">
  <head>
    <meta charset="utf-8">
    <title>作用域</title>
    <script src="js/angular.min.js"></script>
    <script>
      var m1 = angular.module('myApp',[]);
      m1.controller('Aaa',function($scope,$rootScope){
        $scope.name='hello';
        // $scope.age='40';
        // 之后查找的是Aaa中的局部变量
        $rootScope.age='20';
        // 最后查找的是全局变量
      });
      m1.controller('Bbb',function($scope){
        $scope.name='hi';
        // $scope.age='30';
        // 首先查找的是Bbb中的局部变量
      });
    </script>
  </head>
  <body>
    <div ng-controller="Aaa">
      <p>{{name}}</p>
      <div ng-controller="Bbb">
        <p>{{name}}</p>
        <p>{{age}}</p>
      </div>
    </div>
  </body>
</html>
```

## 依赖注入

### 正常的传参

```
function Aaa(n){
  alert(123);
}
Aaa(11);
```

ng-controller="Aaa" 没有传递实参

```
m1.controller('Aaa',function($scope,$rootScope)
```

是在angular内部进行调用，在调用的时候自动 依赖注入进去了。

这就是依赖注入的概念。不是我们自己写参数，而是自动帮我们注入了。

那怎么才能获取到想要的形参呢？

那就要根据写的形参才决定了！

你写一个\$scope，那就等到了了一个局部的作用域的对象。

你写一个\$rootScope，那就等到了一个全局的对象。

参数是不能变的，不然angular就不知道你这个形参要做什么事情？

在angular中，这些注入的东西呢！有统一的名称就做 服务。

服务 以\$开头、后面跟着一个名字。可以实现特有的功能、服务与我们实际开发的。

## 指令系统

### 案例

```
<!DOCTYPE html>
<html ng-app="myApp">
  <head>
    <meta charset="{CHARSET}">
    <title></title>
    <script src="js/angular.min.js"></script>
    <script>
      var m1 = angular.module('myApp',[]);
      m1.controller('Aaa',function($scope){
        $scope.name='hello';
      });
    </script>
  </head>
  <body>
    <div ng-controller="Aaa">
      <p>{{name}}</p>
    </div>
  </body>
</html>
```

### ng-开头的是AngularJS的指令

**ng-controller:** 是控制器连接视图和数据的。

**ng-app:** 初始化指令 从哪里开始执行 AngularJS

**ng-click :** 点击指令

可以html标签中加载

也可以在局部加载

一个大型网站不一定全部用AngularJS开发~可能是一块用

**<html ng-app="myApp">**放在html标签中会整个页面加载、方便我们调用！

后面课程中也会有**自定义指令**

## 双向数据绑定

## MVVM设计模式

M是数据 、V是视图

### 案例

```
<!DOCTYPE html>
<html ng-app="myApp">
  <head>
    <meta charset="{CHARSET}">
    <title></title>
    <script src="js/angular.min.js"></script>
    <script>
      var m1 = angular.module('myApp',[]);
      m1.controller('Aaa',function($scope){
        $scope.name='hello';
      });
    </script>
  </head>
  <body>
    <div ng-controller="Aaa">
      <input type="text" ng-model='name'>
      <!-- ng-model='name'是数据，相当于输入框一个值绑定了数据 -->
      <p>{{name}}</p>
    </div>
  </body>
</html>
```

M: \$scope.name='hello';首先在页面中影响了V: <input type="text" ng-model='name'>

当我们改变V: <input type="text" ng-model='name'>的内容

又会改变M: \$scope.name='hello';

当我们的数据发生改变的时候让视图改变

当视图改变数据改变

效果演示：延时2秒变化内容

### \$timeout

### 延时器

```
<!DOCTYPE html>
<html ng-app="myApp">
  <head>
    <meta charset="{CHARSET}">
    <title></title>
    <script src="js/angular.min.js"></script>
    <script>
      var m1 = angular.module('myApp',[]);
      m1.controller('Aaa',function($scope,$timeout){
        $scope.name='hello';
        // setTimeout(function(){
        //   $scope.name='h1';
        // },2000);
        // setTime方法
        // 要用angular提供的服务$timeout
        // $timeout(function(){
        //   $scope.name='h1';
        // },2000);
        $scope.show = function(){
```

```

        $scope.name='hi';
    }
    });
</script>
</head>
<body>
    <!-- <div ng-controller="Aaa" ng-click="name='hi'">. -->
    <div ng-controller="Aaa" ng-click="show()">
        <p>{{name}}</p>
    </div>
</body>
</html>

```

## \$currency

### 过滤器

### 案例

```

<!DOCTYPE html>
<html ng-app="myApp">
    <head>
        <meta charset="utf-8">
        <title>购物车</title>
        <script src="js/angular.min.js"></script>
        <script>
            var m1 = angular.module('myApp',[]);
            m1.controller('Aaa',function($scope,$timeout){
                $scope.iphone = {
                    money : 5,
                    num : 1,
                    fre : 10
                };
                $scope.sum = function(){
                    return $scope.iphone.money * $scope.iphone.num
                }
            });

        </script>
    </head>
    <body>
        <div ng-controller="Aaa">
            <p>价格: <input type="text" ng-model="iphone.money"></p>
            <p>个数: <input type="text" ng-model="iphone.num"></p>
            <!-- <p>费用: <span>{{ iphone.money * iphone.num | currency:'¥' }}</span></p> -->
            <p>费用: <span>{{ sum() | currency:'¥' }}</span></p>
            <p>运费: <span>{{ iphone.fre | currency:'¥' }}</span></p>
            <p>合计: <span>{{ sum() + iphone.fre | currency:'¥' }}</span></p>
        </div>
    </body>
</html>

```

## \$watch



## 监听

监听相关数据变化，做出相应处理。

接受三个参数，前两个是必选的、第三个是可选的。

```
$scope.$watch("iphone.money",function())
```

```
$scope.$watch("iphone.money",function() {
    console.log(123);
})
```

可以监听iphone.money的动态。

```
$scope.$watch("iphone",function() {
    console.log(123);
},true)
```

监听iphone这个整体，需要第三个参数true

深度监听：可以监听整体（集合）

```
$scope.Swatch("iphone.money",function(newVal,oldVal) {
    console.log(new Val);
    console.log(oldVal);
},true)
```

监听新的值、老的值 newVal,oldVal

## 也可以监听函数

```
$scope.sum = function(){
    return $scope.iphone.money * $scope.iphone.num
}
```

```
$scope.$watch($scope.sum,function(newVal,oldVal){
    console.log(newVal);
    console.log(oldVal);
},true)
```

监听的是return返回的结果

## 案例（完整版）

```
<!DOCTYPE html>

<html ng-app="myApp">

  <head>

    <meta charset="utf-8">

    <title>购物车</title>

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

    <script>

      var m1 = angular.module('myApp',[]);

      m1.controller('Aaa',function($scope,$timeout){

        $scope.iphone = {

          money : 5,

          num : 1,

          fre : 10

        };

        $scope.sum = function(){

          return $scope.iphone.money * $scope.iphone.num

        }

        $scope.$watch($scope.sum,function(newVal,oldVal){

          // console.log(newVal);

          // console.log(oldVal);
```

```
$scope.iphone.fre = newVal >= 100 ? 0 : 10
    }, true)
});

</script>
</head>
<body>
    <div ng-controller="Aaa">
        <p>价格: <input type="text" ng-model="iphone.money"></p>
        <p>个数: <input type="text" ng-model="iphone.num"></p>
        <!-- <p>费用: <span>{{ iphone.money * iphone.num | currency:'¥' }}</span></p> -->
        <p>费用: <span>{{ sum() | currency:'¥' }}</span></p>
        <p>运费: <span>{{ iphone.fre | currency:'¥' }}</span></p>
        <p>合计: <span>{{ sum() + iphone.fre | currency:'¥' }}</span></p>
    </div>
</body>
</html>
```