第二讲 html5 框架+Crosswalk 打包 app 以及 Angularjs 基础（初步认识了解 Angularjs）

学习要点：

1. html5 框架+Crosswalk 打包 app
2. 什么是 angularjs
3. Angularjs 之前问什么要学 ionic
4. 开发工具介绍以及 Hello Angular
5. Angularjs 中常用名词 也就是所说的常用**指令**
6. Angularjs 表达式
7. AngularJS 控制器
8. AngularJS $http 请求数据
9. AngularJS 过滤器
10. AngularJS 模块

主讲：（树根）

合作网站：www.phonegap100.com (PhoneGap 中文网)

合作网站：www.itying.com (IT 营)

1. html5 框架+Crosswalk 打包 app

2. 什么是 Angularjs

AngularJS 最初由 Misko Hevery 和 Adam Abrons 于 2009 年开发，后来成为了 Google 公司的项目。AngularJS 弥补了 HTML 在构建应用方面的不足，其通过使用标识符（directives）结构，来扩展 Web 应用中的 HTML 词汇，使开发者可以使用 HTML 来声明动态内容，从而使得 Web 开发和测试工作变得更加容易。

Misko Hevery

Angularjs 版本简介

https://github.com/angular/angular.js/releases/

AngularJS 功能：

AngularJS 是专门为应用程序设计的 HTML。

AngularJS 使得开发现代的单一页面应用程序（SPAs：Single Page Applications）变得更加容易。

1 AngularJS 把应用程序数据绑定到 HTML 元素。
2 AngularJS 可以克隆和重复 HTML 元素。
3 AngularJS 可以隐藏和显示 HTML 元素。
4 AngularJS 可以在 HTML 元素"背后"添加代码。
5 AngularJS 支持输入验证

Angularjs 号称 下一代 web 应用 主要特性如下：

1.MVC
2.模块化与依赖注入
3.双向数据绑定
4.指令与 UI 控件

3. 学 Angularjs 之前问什么要学 ionic

1. **AngularJs integrate-整合了 AngularJs**

2. **Url routing，use AngularUI Router**

url 路由使用 AngularUI Router，可以指定不同的路由，方便开发和集成

3 . **AngularJS Extensions & Directives 扩展了 AngularJS 指令**

ion-tab, ion-content, ion-nav-view, ion-header
$ionicPopup,$ionicLoading, $ionicModal…

4. Hello Angular

Angularjs 资源：

http://Angularjs.org 官方网站正常打不开 但是打不开 大家都懂的
http://www.angularjs.cn/
http://docs.angularjs.cn/api
http://www.ngnice.com/
https://github.com/angular

Angularjs 下载：

http://www.bootcdn.cn/angular.js/
通过 nodejs 下载：npm install angular

为了使用 Angular，所有应用都必须首先做两件事情

1. 下载加载 angular.js 库
2. 使用 ng-app 指令告诉 angular 应该管理 DOM 中的哪一些部分

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>无标题文档</title>
        <script type="text/javascript" src="angular-1.3.0.js"></script>
    </head>
    <body>
        <div ng-app="">
          <p>在输入框中尝试输入：</p>
          <p>姓名：<input type="text" ng-model="name"></p>
          <p ng-bind="name"></p>
        </div>

    </body>
</html>
```

5. Angularjs 中常用名词 也就是所说的常用指令

HTML5 允许扩展的（自制的）属性，以 **data-** 开头。
AngularJS 属性以 **ng-** 开头，但是您可以使用 **data-ng-** 来让网页对 HTML5 有效

俗话说 下面的指令可以在开头加上 data- 例如 ng_app 等同于 **data _ng_app**

指令	描述	讲解
ng_app	定义应用程序的根元素。	指令
ng_bind	绑定 HTML 元素到应用程序数据。	简介
ng_click	定义元素被单击时的行为。	HTML 事件
ng_controller	为应用程序定义控制器对象。	控制器

ng_disabled	绑定应用程序数据到 HTML 的 disabled 属性。	HTML DOM
ng_init	为应用程序定义初始值。	指令
ng_model	绑定应用程序数据到 HTML 元素。	指令
ng_repeat	为控制器中的每个数据定义一个模板。	指令
ng_show	显示或隐藏 HTML 元素。	HTML DOM

1. ng_app ng_bind ng_model {{}}案例演示

ng_app：

ng-app 指令定义了 AngularJS 应用程序的 根元素。

ng-app 指令在网页加载完毕时会自动引导（自动初始化）应用程序。

稍后您将学习到 ng-app 如何通过一个值（比如 ng-app="myModule"）连接到代码模块。

ng-model 指令：

ng-model 指令 绑定 HTML 元素 到应用程序数据。

ng-model 指令也可以：

为应用程序数据提供类型验证（number、email、required）。
为应用程序数据提供状态（invalid、dirty、touched、error）。
为 HTML 元素提供 CSS 类。
绑定 HTML 元素到 HTML 表单。

ng_bind 指令 等同于{{}}

绑定 HTML 元素到应用程序数据。

示例 1：

```
<!DOCTYPE html>
<html>
<body>
<div ng-app="">
```

```
  <p>在输入框中尝试输入：</p>
  <p>姓名：<input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>
<script src="angular.min.js"></script>

</body>
</html>
```

示例 2：{{}}等同于 ng_bind

```
<!DOCTYPE html>
<html>
<body>
<div ng-app="">
  <p>在输入框中尝试输入：</p>
  <p>姓名：<input type="text" ng-model="name"></p>
  <p>{{name}}</p>
</div>
<script src="angular.min.js"></script>

</body>
</html>
```

实例讲解：

当网页加载完毕，AngularJS 自动开启。

ng-app 指令告诉 AngularJS， 元素是 AngularJS **应用程序** 的"所有者"。

ng-model 指令把输入域的值绑定到应用程序变量 **name**。

ng-bind 指令把应用程序变量 name 绑定到某个段落的 innerHTML。

2. ng_init

ng-init 指令
ng-init 指令为 AngularJS 应用程序定义了 初始值。

通常情况下，不使用 ng-init。您将使用一个控制器或模块来代替它。

稍后您将学习更多有关控制器和模块的知识。

```
<div ng-app="" ng-init="firstName='John'">

<p>姓名为 <span ng-bind="firstName"></span></p>

</div>
```

3.data-指令 data-ng-init 与 ng-init 等价

```
<div data-ng-app="" data-ng-init="firstName='John'">

<p>姓名为 <span data-ng-bind="firstName"></span></p>

</div>
```

6. Angularjs 表达式

AngularJS 表达式写在双大括号内：**{{ expression }}**。

AngularJS 表达式把数据绑定到 HTML，这与 **ng-bind** 指令有异曲同工之妙。

AngularJS 将在表达式书写的位置"输出"数据。

AngularJS 表达式 很像 **JavaScript 表达式**：它们可以包含文字、运算符和变量。

实例 {{ 5 + 5 }} 或 {{ firstName + " " + lastName }}

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="">
  <p>我的第一个表达式： {{ 5 + 5 }}</p>
```

```
</div>

<script src="angular.min.js"></script>
</body>
</html>
```

AngularJS 数字

```
<div ng-app="" ng-init="quantity=1;cost=5">

<p>总价： {{ quantity * cost }}</p>

</div>
```

AngularJS 字符串

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>姓名： {{ firstName + " " + lastName }}</p>

</div>
```

AngularJS 对象

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>姓为 {{ person.lastName }}</p>

</div>
```

AngularJS 数组

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">

<p>第三个值为 {{ points[2] }}</p>

</div>
```

7. Angularjs 控制器

AngularJS 控制器 控制 AngularJS 应用程序的数据。

AngularJS 控制器是常规的 JavaScript 对象。

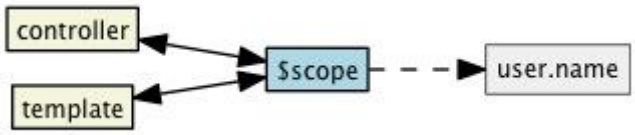
AngularJS 应用程序被控制器控制。

ng-controller 指令定义了应用程序控制器。

控制器是 JavaScript 对象，由标准的 JavaScript 对象的构造函数 创建。

控制器的 $scope 是控制器所指向的应用程序 HTML 元素。

angular 中$scope 是连接 controllers(控制器)和 templates(模板 view/视图)的主要胶合体。我们可以把我们的 model 存放在 scope 上，来达到双向你绑定。




```
<!DOCTYPE html>
<html>
<body>
<div ng-app="">
<div ng-controller="personController">
名: <input type="text" ng-model="person.firstName"><br>
姓: <input type="text" ng-model="person.lastName"><br>
<br>
姓名: {{person.firstName + " " + person.lastName}}
</div>
</div>
<script>
function personController($scope) {     //不建议这样写
    $scope.person = {
        firstName: "John",
        lastName: "Doe"
    };
}
</script>
<script src="angular.min.js"></script>
</body>
</html>
```

实例讲解：

AngularJS 应用程序由 **ng-app** 定义。应用程序在 内运行。

ng-controller 指令把控制器命名为 **object**。

函数 **personController** 是一个标准的 JavaScript **对象的构造函数**。

控制器对象有一个属性：**$scope.person**。

person 对象有两个属性：**firstName** 和 **lastName**。

ng-model 指令绑定输入域到控制器的属性（firstName 和 lastName）。

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="" ng-controller="personController">

名: <input type="text" ng-model="person.firstName"><br>
姓: <input type="text" ng-model="person.lastName"><br>
<br>
姓名: {{person.fullName()}}

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe",
        fullName: function() {
            var x = $scope.person;
            return x.firstName + " " + x.lastName;
        }
    };
}
</script>
```

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


</body>
</html>
```

ng-repeat 指令结合 ng-controller

```
<div ng-app="" ng-controller="namesController">

<ul>
  <li ng-repeat="x in names">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>

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

<script>
function namesController($scope) {
    $scope.names = [
        {name:'Jani',country:'Norway'},
        {name:'Hege',country:'Sweden'},
        {name:'Kai',country:'Denmark'}
    ];
}
</script>
```

8. Angularjs $http 请求数据

1. get 请求

```
<div ng-app="" ng-controller="customersController">

<ul>
  <li ng-repeat="x in names">
    {{ x.Name + ', ' + x.Country }}
  </li>
</ul>

</div>

<script>
function customersController($scope,$http) {
    $http.get("http://www.w3cschool.cc/try/angularjs/data/Customers_JSON.php")
    .success(function(response) {$scope.names = response;});
}
</script>
```

$http get 实例 1：

```
$http.get("http://www.w3cschool.cc/try/angularjs/data/Customers_JSON.php").success(function
(response) {$scope.names = response;});
```

$http get 实例 2：

```
$http.get(url,{params:{id:'5'}}) .success(function(response) {
            $scope.names = response;
            }).error(function(data){

            //错误代码

            });
```

$http post 实例：

```
var postData={text:'这是 post 的内容'};
        var config={params:{id:'5'}}
        $http.post(url,postData,config) .success(function(response) {
            $scope.names = response;
        }).error(function(data){
            //错误代码
```

```
        });
```

$http　Jsonp 实例：

```
    myUrl = "http://www.phonegap100.com/appapi.php?a=getPortalList&catid=20&page=1&callback=JSON_CALLBACK";
    $http.jsonp(myUrl).success(
        function(data){

            $scope.portalcate = data.result;
        }
    ).error(function(){
        alert('shibai');

    });
```

9. Angularjs　过滤器

AngularJS 过滤器

AngularJS 过滤器可用于转换数据：

过滤器	描述
currency	格式化数字为货币格式。
filter	从数组项中选择一个子集。
lowercase	格式化字符串为小写。
orderBy	根据某个表达式排列数组。
uppercase	格式化字符串为大写。

向表达式添加过滤器

过滤器可以通过一个管道字符（|）和一个过滤器添加到表达式中。

（下面的两个实例，我们将使用前面章节中提到的 person 控制器）

uppercase 过滤器格式化字符串为大写：

```
<div ng-app="" ng-controller="personController">

<p>姓名为 {{ person.lastName | uppercase }}</p>

</div>
```

lowercase 过滤器格式化字符串为小写：

```
<div ng-app="" ng-controller="personController"">

<p>姓名为 {{ person.lastName | lowercase }}</p>

</div>
```

currency 过滤器
currency 过滤器格式化数字为货币格式：

```
<div ng-app="" ng-controller="costController">
数量：<input type="number" ng-model="quantity">
价格：<input type="number" ng-model="price">
<p>总价 = {{ (quantity * price) | currency }}</p>
</div>
```

向指令添加过滤器
过滤器可以通过一个管道字符（|）和一个过滤器添加到指令中。

orderBy 过滤器根据某个表达式排列数组：

```
<div ng-app="" ng-controller="namesController">
<p>循环对象：</p>
<ul>
  <li ng-repeat="x in names | orderBy:'country'">
```

```
    {{ x.name + ', ' + x.country }}
  </li>
</ul>
<div>
```

过滤输入

输入过滤器可以通过一个管道字符（|）和一个过滤器添加到指令中，该过滤器后跟一个冒号和一个模型名称。

filter 过滤器从数组中选择一个子集：

```
<div ng-app="" ng-controller="namesController">
<p>输入过滤：</p>
<p><input type="text" ng-model="name"></p>
<ul>
  <li ng-repeat="x in names | filter:name | orderBy:'country'">
    {{ (x.name | uppercase) + ', ' + x.country }}
  </li>
</ul>
</div>
```

10.Angularjs 模块

1. 为什么要使用模块

控制器污染了全局命名空间

http://baike.baidu.com/view/4174721.htm

本教程中，截至目前为止的所有实例都使用了全局函数。

在所有的应用程序中，都应该尽量避免使用全局变量和全局函数。

全局值（变量或函数）可被其他脚本重写或破坏。

为了解决这个问题，AngularJS 使用了模块。

2. 普通的控制器 和 AngularJS 模块

AngularJS 普通的控制器

```
<!DOCTYPE html>
<html>
<body>
<div ng-app="" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script>
function myCtrl($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
}
</script>
<script src="angular.min.js"></script>

</body>
</html>
```

使用一个由 模块 替代的控制器：

```
<!DOCTYPE html>

<html>

<head>

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

</head>



<body>



<div ng-app="myApp" ng-controller="myCtrl">

{{ firstName + " " + lastName }}

</div>
```

```
<script>
var app = angular.module("myApp", []);

app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>
</body>
</html>
```

3. AngularJS 应用程序文件

现在您已经知道模块是什么以及它们是如何工作的，现在您可以尝试创建您自己的应用程序文件。

您的应用程序至少应该有一个模块文件，一个控制器文件。

首先，创建模块文件 "myApp.js"：

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

然后，创建控制器文件。本实例中是 "myCtrl.js"：

```
app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
```

```
    $scope.lastName = "Doe";

});
```

最后，编辑 HTML 引入模块：

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

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

<script src="myApp.js"></script>
<script src="myCtrl.js"></script>

</body>
</html>
```