**AngularJs**

## 一、AngularJS 概述

**1 AngularJS 是一个 JavaScript 框架。**它可通过 <script> 标签添加到 HTML 页面。通过 指令 扩展了 HTML，且通过 表达式 绑定数据到 HTML。

AngularJS 通过 ng-directives 扩展了 HTML。

ng-app 指令定义一个 AngularJS 应用程序。

ng-model 指令把元素值（比如输入域的值）绑定到应用程序。

ng-bind 指令把应用程序数据绑定到 HTML 视图。

**2 四大特点**

**（1）MVC模式：**

　@Model:数据,其实就是angular变量($scope.XX,$rootScope.XX);

　@View:数据的呈现,Html+Directive(指令);

　@Controller:操作数据,就是function,数据的增删改查;

**（2）双向绑定:**

方向一:Model--->View

{{Model数据}} 或<XXX ng-xxx="Model数据"> Model变View跟着变;

方向二:View--->Model

<表单控件 ng-model="Model数据名"> View变Model跟着变;

**（3）依赖注入:**

依赖注入(Dependency Injection,简称DI)是一种设计模式, 指某个对象依赖的其他对象无需手工创建，只需要“吼一嗓子”，则此对象在创建时，其依赖的对象由框架来自动创建并注入进来,其实就是最少知识法则;模块中所有的service和provider两类对象，都可以根据形参名称实现DI.

myModule.factory('$alert', function($window) {

return {

alert: function(text) {

$window.alert(text);

}

};

});

var myController = function($scope, $alert) {

$scope.message = function(msg) {

console.log(msg);

$alert.alert(msg);

};

};

myController.$inject = ['$scope', '$alert'];

**（4）模块化设计:高内聚低耦合法则,**

1)官方提供的模块 ng、ngRoute、ngAnimate、ngTouch

2)用户自定义的模块 angular.module('模块名',[ ])

## 二、开发总结

**1 引入文件**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>

</head>

<body>

<div ng-app="">

<p>名字 : <input type="text" ng-model="name"></p>

<h1>Hello {{name}}</h1>

</div>

</body>

</html>

**2 基础表达式**

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

名: <input type="text" ng-model="firstName"><br>

姓: <input type="text" ng-model="lastName"><br>

<br>

姓名: {{firstName + " " + lastName}}

</div>

<script>

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

app.controller('myCtrl', function($scope)

{

$scope.firstName= "John";

$scope.lastName= "Doe";

});

</script>

**3 表达式**

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

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

</div>

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

<p>总价： {{ quantity \* cost }}</p>

</div>

**4 指令**

AngularJS 指令是扩展的 HTML 属性，带有前缀 ng-。

ng-app 指令初始化一个 AngularJS 应用程序。

ng-init 指令初始化应用程序数据。

ng-model 指令把元素值（比如输入域的值）绑定到应用程序。

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

<p>在输入框中尝试输入：</p>

<p>姓名：<input type="text" ng-model="firstName"></p>

<p>你输入的为： {{ firstName }}</p>

</div>

**5 模型**

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

名字: <input ng-model="name">

</div>

<script>

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

app.controller('myCtrl', function($scope)

{

$scope.name = "John Doe";

});

</script>

**6 双向绑定**

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

名字: <input ng-model="name">

<h1>你输入了: {{name}}</h1>

</div>

**7 Scope(作用域)**

是应用在 HTML (视图) 和 JavaScript (控制器)之间的纽带。

Scope 是一个对象，有可用的方法和属性。

Scope 可应用在视图和控制器上。

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

<input ng-model="name">

<h1>{{greeting}}</h1>

<button ng-click='sayHello()'>点我</button>

</div>

<script>

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

app.controller('myCtrl', function($scope)

{

$scope.name = "Runoob";

$scope.sayHello = function()

{

$scope.greeting = 'Hello ' + $scope.name + '!';

};

});

</script>

**8 控制器**

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。

AngularJS 应用程序由 ng-app 定义。应用程序在 <div> 内运行。

ng-controller="myCtrl"属性是一个 AngularJS 指令。用于定义一个控制器。

myCtrl 函数是一个 JavaScript 函数。

AngularJS 使用$scope 对象来调用控制器。

在 AngularJS 中， $scope 是一个应用对象(属于应用变量和函数)。

控制器的 $scope （相当于作用域、控制范围）用来保存AngularJS Model(模型)的对象。

控制器在作用域中创建了两个属性 (firstName 和 lastName)。

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

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

名: <input type="text" ng-model="firstName"><br>

姓: <input type="text" ng-model="lastName"><br>

<br>

姓名: {{firstName + " " + lastName}}

</div>

<script>

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

app.controller('myCtrl', function($scope)

{

$scope.firstName = "John";

$scope.lastName = "Doe";

});

</script>

**9 服务**

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

app.controller('myCtrl', function($scope, $http)

{

$http.get("welcome.htm").then(function (response)

{

$scope.myWelcome = response.data;

});

});

$location.absUrl();

$timeout(function ()

{

$scope.myHeader = "How are you today?";

}, 2000);

$interval(function ()

{

$scope.theTime = new Date().toLocaleTimeString();

}, 1000);

$http({

method: 'GET',

url: 'https://www.runoob.com/try/angularjs/data/sites.php'

}).then(function successCallback(response)

{

$scope.names = response.data.sites;

}, function errorCallback(response)

{

// 请求失败执行代码

});

**10 select 框架**。

<select>

<select ng-model="selectedSite">

<option ng-repeat="x in sites" value="{{x.url}}">{{x.site}}</option>

</select>

<h1>你选择的是: {{selectedSite}}</h1>

**11 表格**

<table>

<tr ng-repeat="x in names">

<td>{{ $index + 1 }}</td>

<td>{{ x.Name }}</td>

<td>{{ x.Country }}</td>

</tr>

</table>

<table>

<tr ng-repeat="x in names">

<td ng-if="$odd" style="background-color:#f1f1f1">{{ x.Name }}</td>

<td ng-if="$even">{{ x.Name }}</td>

<td ng-if="$odd" style="background-color:#f1f1f1">{{ x.Country }}</td>

<td ng-if="$even">{{ x.Country }}</td>

</tr>

</table>

**12 执行 SQL 获取数据**

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

<table>

<tr ng-repeat="x in names">

<td>{{ x.Name }}</td>

<td>{{ x.Country }}</td>

</tr>

</table>

</div>

<script>

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

app.controller('customersCtrl', function($scope, $http)

{

$http.get("http://www.runoob.com/try/angularjs/data/Customers\_SQL.aspx")

.success(function (response) {$scope.names = response.records;});

});

</script>

**13 事件**

**ng-show 指令可用于设置应用中的一部分是否可见 。**

**ng-show="false" 可以设置 HTML 元素 不可见。**

**ng-show="true" 可以以设置 HTML 元素可见。**

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

<button ng-click="toggle()">隐藏/显示</button>

<p ng-hide="myVar">

名: <input type="text" ng-model="firstName"><br>

姓名: <input type="text" ng-model="lastName"><br>

<br>

Full Name: {{firstName + " " + lastName}}

</p>

</div>

<script>

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

app.controller('personCtrl', function($scope)

{

$scope.firstName = "John",

$scope.lastName = "Doe"

$scope.myVar = false;

$scope.toggle = function()

{

$scope.myVar = !$scope.myVar;

};

});

</script>

**14 模块**

模块是应用程序中不同部分的容器。

模块是应用控制器的容器。

控制器通常属于一个模块。

<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>

**15 表单**

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

<form novalidate>

First Name:<br>

<input type="text" ng-model="user.firstName"><br>

Last Name:<br>

<input type="text" ng-model="user.lastName">

<br><br>

<button ng-click="reset()">RESET</button>

</form>

<p>form = {{user}}</p>

<p>master = {{master}}</p>

</div>

<script>

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

app.controller('formCtrl', function($scope)

{

$scope.master = {firstName: "John", lastName: "Doe"};

$scope.reset = function()

{

$scope.user = angular.copy($scope.master);

};

$scope.reset();

});

</script>

ng-app 指令定义了 AngularJS 应用。

ng-controller 指令定义了应用控制器。

ng-model 指令绑定了两个 input 元素到模型的 user 对象。

formCtrl 函数设置了 master 对象的初始值，并定义了 reset() 方法。

reset() 方法设置了 user 对象等于 master 对象。

ng-click 指令调用了 reset() 方法，且在点击按钮时调用。

novalidate 属性在应用中不是必须的，但是你需要在 AngularJS 表单中使用，用于重写标准的 HTML5 验证。

**16 输入验证**

<!DOCTYPE html>

<html>

<script src="http://apps.bdimg.com/libs/angular.js/1.4.6/angular.min.js"></script>

<body>

<h2>Validation Example</h2>

<form ng-app="myApp" ng-controller="validateCtrl"

name="myForm" novalidate>

<p>用户名:<br>

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

<span style="color:red" ng-show="myForm.user.$dirty && myForm.user.$invalid">

<span ng-show="myForm.user.$error.required">用户名是必须的。</span>

</span>

<input type="text" ng-minlength="5" ng-maxlength="20" ng-pattern="/[a-zA-Z]/" />

<input type="number" name="age" ng-model="user.age" />

<input type="url" name="homepage" ng-model="user.facebook\_url" />

</p>

<p>邮箱:<br>

<input type="email" name="email" ng-model="email" required>

<span style="color:red" ng-show="myForm.email.$dirty && myForm.email.$invalid">

<span ng-show="myForm.email.$error.required">邮箱是必须的。</span>

<span ng-show="myForm.email.$error.email">非法的邮箱。</span>

</span>

</p>

<p>

<input type="submit"

ng-disabled="myForm.user.$dirty && myForm.user.$invalid ||

myForm.email.$dirty && myForm.email.$invalid">

</p>

</form>

<script>

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

app.controller('validateCtrl', function($scope)

{

$scope.user = 'John Doe';

$scope.email = 'john.doe@gmail.com';

});

</script>

</body>

</html>

**17 API**

angular.lowercase() 转换字符串为小写

angular.uppercase() 转换字符串为大写

angular.isString() 判断给定的对象是否为字符串，如果是返回 true。

angular.isNumber() 判断给定的对象是否为数字，如果是返回 true。

**18 Bootstrap**

AngularJS 的首选样式表是 Twitter Bootstrap， Twitter Bootstrap 是目前最受欢迎的前端框架。

<link

rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">

**19 依赖注入**

AngularJS 提供很好的依赖注入机制。以下5个核心组件用来作为依赖注入：

@value

@factory

@service

@provider

@constant

var mainApp = angular.module("mainApp", []);

mainApp.value("defaultInput", 5);

mainApp.factory('MathService', function()

{

var factory = {};

factory.multiply = function(a, b)

{

return a \* b

}

return factory;

});

mainApp.service('CalcService', function(MathService)

{

this.square = function(a)

{

return MathService.multiply(a,a);

}

});

**20 路由**

AngularJS 路由允许我们通过不同的 URL 访问不同的内容。

通过 AngularJS 可以实现多视图的单页Web应用（single page web application，SPA）。

通常我们的URL形式为 http://runoob.com/first/page，但在单页Web应用中 AngularJS 通过 # + 标记 实现，例如：

http://runoob.com/#/first

http://runoob.com/#/second

http://runoob.com/#/third

当我们点击以上的任意一个链接时，向服务端请的地址都是一样的 (http://runoob.com/)。 因为 # 号之后的内容在向服务端请求时会被浏览器忽略掉。

所以我们就需要在客户端实现 # 号后面内容的功能实现。 AngularJS 路由 就通过 # + 标记 帮助我们区分不同的逻辑页面并将不同的页面绑定到对应的控制器上。

<html>

<head>

<meta charset="utf-8">

<title>AngularJS 路由实例 - 菜鸟教程</title>

</head>

<body ng-app='routingDemoApp'>

<h2>AngularJS 路由应用</h2>

<ul>

<li><a href="#/">首页</a></li>

<li><a href="#/computers">电脑</a></li>

<li><a href="#/printers">打印机</a></li>

<li><a href="#/blabla">其他</a></li>

</ul>

<div ng-view></div>

<script src="http://apps.bdimg.com/libs/angular.js/1.4.6/angular.min.js"></script>

<script src="http://apps.bdimg.com/libs/angular-route/1.3.13/angular-route.js"></script>

<script>

angular.module('routingDemoApp',['ngRoute'])

.config(['$routeProvider', function($routeProvider)

{

$routeProvider

.when('/',{template:'这是首页页面'})

.when('/computers',{template:'这是电脑分类页面'})

.when('/printers',{template:'这是打印机页面'})

.otherwise({redirectTo:'/'});

}]);

</script>

</body>

</html>

**21 综合应用**

<html ng-app="myNoteApp">

<head>

<meta charset="utf-8">

<script src="http://apps.bdimg.com/libs/angular.js/1.4.6/angular.min.js"></script>

</head>

<body>

<div ng-controller="myNoteCtrl">

<h2>我的笔记</h2>

<p><textarea ng-model="message" cols="40" rows="10"></textarea></p>

<p>

<button ng-click="save()">保存</button>

<button ng-click="clear()">清除</button>

</p>

<p>Number of characters left: <span ng-bind="left()"></span></p>

</div>

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

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

</body>

</html>

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

app.controller("myNoteCtrl", function($scope)

{

$scope.message = "";

$scope.left = function() {return 100 - $scope.message.length;};

$scope.clear = function() {$scope.message = "";};

$scope.save = function() {alert("Note Saved");};

});

以上实例是一个完整的 AngularJS 单页Web应用（single page web application，SPA）。

<html> 元素包含了 AngularJS 应用 (ng-app=)。

<div> 元素定义了 AngularJS 控制器的作用域 (ng-controller=)。

在一个应用可以由很多控制器。

应用文件(my...App.js) 定义了应用模型代码。

一个或多个控制器文件 (my...Ctrl.js) 定义了控制器代码。

ng-app 指令位于应用的根元素下。

对于单页Web应用（single page web application，SPA），应用的根通常为 <html> 元素。

一个或多个 ng-controller 指令定义了应用的控制器。每个控制器有他自己的作用域：: 定义的 HTML 元素。

AngularJS 在 HTML DOMContentLoaded 事件中自动开始。如果找到 ng-app 指令 ， AngularJS 载入指令中的模块，并将 ng-app 作为应用的根进行编译。

应用的根可以是整个页面，或者页面的一小部分，如果是一小部分会更快编译和执行。

## 三、优点

**1 AngularJS是一套完整的框架**，angular有自带的数据绑定、render渲染、angularUI库,过滤器,directive(模板),服务q(defer),http，inject(依赖注入),

factory,provider……，等等一系列工具，基本上只要你在做web开发用过的东西，它都有一个。但是这些东西React自身都没有。

**2 angularjs的架构清晰，**分工明确，扩展性良好，model，view，controller谁在什么时候做什么事情说的很清楚，angular能够让程序员真正专注于业务逻辑，

而且因为对html侵入不大，非常易于和designer协作。整个框架充满了DI的思路，耦合性非常低，对象都是被inject的，也就是说每个对象都可以轻易被替换而

不影响其他对象。

**3 Angular生产效率高**，单向数据流什么的想法非常好，但是写起来太麻烦！我只想变更个很简单的数据还要经过action、dispatcher、reduce、view四步

，angular里一行代码就搞定的事情在react里却如此麻烦

## 四、缺点

**1 性能 ，**双向数据绑定是一把双刃剑。随着组件增加，项目越来越复杂，双向数据绑定带来性能问题。双向数据绑定是如何影响性能的？在JavaScript（ES5）中，并没有实现当变量或对象改变时发出通知的功能，Angular的实现方法被叫做“Dirty-checking（脏检查机制），通过跟踪数据的改变再动态更新用户界面（UI）。在Angular的作用域中任何操作的执行都会引发Dirty-checking，随着绑定数量的增加性能就会越低。

**2 Angular 2.0推翻重做使得目前不宜采用此框架**

Angular 1.x版本其实是个比较旧的东西了，现在看来有些理念过时了，比如依赖注入、自己独特的模块化，这些东西其实在ES6下已经很好地被解决了。 Angular的2.0几乎是一个推翻重做的框架，估计不会有1.X的upgrade方案。所以如果现在新开始的项目采用Angular的话，会是一个很尴尬的时机。同样，如此大的改动似乎也反面印证了1.X并不是那么好。

## 五、使用场景

1 表单参数绑定

2 模块化

3 web管理系统