



آموزش

AngularJS

www.tahlildadeh.com

سورس سرا
آموزش برنامه نویسی



نویسنده: مهندس افشین رفوآ

برای دانلود سورس پروژه ها به سایت www.tahlildadeh.com مراجعه فرمائید.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

کتاب آموزش گام به گام **AngularJS** به زبان فارسی

مولف مهندس افشین رفوآ

تقدیم به نائب امام عصر، آیت الله خامنه ای
که عصا زدنش ضرب آهنگ حیدری دارد.

تقدیم به همه جویندگان علم که توان و امکان شرکت در
کلاس های حضوری ما را ندارند.

۹	مقدمه
۹	توجه :
۱۰	آموزش کتابخانه ی AngularJS
۱۰	نمونه ای از کاربرد AngularJS :
۱۰	آنچه که باید از قبل بدانید:
۱۱	پیشینه ی چارچوب کاری AngularJS
۱۱	مقدمه ای بر AngularJS
۱۲	مثالی از AngularJS
۱۲	Directive های AngularJS
۱۳	عبارت ها (expression) AngularJS
۱۳	برنامه های (application) AngularJS
۱۴	عبارت ها/Expression در AngularJS
۱۵	اعداد در AngularJS
۱۵	رشته ها در AngularJS
۱۵	اشیا در AngularJS
۱۶	آرایه ها در AngularJS
۱۶	مقایسه عبارات در Angularjs و JavaScript
۱۶	Modules در AngularJS
۱۷	یک ماژول به همراه یک کنترلر
۱۷	قرار دادن ماژول ها و کنترلرها در فایل های جدا
۱۸	توابع می توانند فضاهای نامی سراسری را تخریب کنند
۱۸	زمان بارگذاری کتابخانه
۱۹	Directives ها در AngularJS
۱۹	Angular Directives
۲۰	اتصال داده (Data Binding)
۲۰	تکرار کردن المان های HTML

۲۲ دستور ng-app
۲۲ دستور in-init
۲۲ دستور ng-model
۲۳ دستور ng-repeat
۲۳ دایرکتیو ng-model در AngularJS
۲۳ دایرکتیو ng-model
۲۳ پیوند دو طرفه
۲۴ اعتبارسنجی مقدار input کاربر
۲۴ وضعیت یک برنامه کاربردی (Application Status)
۲۴ کلاس های CSS
۲۵ لیست انتخاب (Dropdown) در AngularJS
۲۵ ساختن یک dropdown با استفاده از ng-options
۲۵ مقایسه ng-repeat با ng-options
۲۶ برای چه باید استفاده کنیم؟
۲۷ منبع داده (Data Source) به صورت یک object
۲۹ Controller ها در AngularJS
۲۹ AngularJS Controllers
۳۰ تشریح برنامه ی فوق
۳۰ متدهای کنترلر
۳۱ تعریف کنترلرها در فایل های خارجی
۳۲ مثال دیگر
۳۲ محدوده (Scope) در AngularJS
۳۳ چگونه از Scope استفاده کنیم؟
۳۳ درک کردن Scope
۳۴ Scope خود را بشناسید
۳۴ هر المان به همان object تکرارشونده دسترسی دارد. در این مورد یک رشته با X نشان داده شده است.
۳۴ Scope ریشه
۳۵ فیلترها در AngularJS

۳۶.....	افزودن فیلتر به عبارت ها
۳۶.....	فیلتر currency
۳۷.....	افزودن فیلتر به Directive ها
۳۷.....	فیلتر کردن ورودی
۳۸.....	سرویس های AngularJS
۳۸.....	سرویس چیست؟
۳۹.....	چرا از سرویس ها استفاده می کنیم؟
۳۹.....	سرویس \$http
۳۹.....	سرویس \$timeout
۴۰.....	سرویس \$interval
۴۰.....	ساختن سرویس شخصی
۴۱.....	استفاده کردن از یک سرویس دلخواه داخل یک فیلتر
۴۲.....	AngularJS و سرویس \$http
۴۲.....	فراهم کردن دادن داده ها
۴۴.....	AngularJS \$http یک سرویس اصلی است که برای خواندن اطلاعات از سرویس دهنده های وب کاربرد دارد.
۴۵.....	جداول در AngularJS
۴۵.....	نمایش دادن داده ها در جدول
۴۶.....	نمایش دادن اطلاعات به وسیله ی استایل CSS
۴۸.....	نمایش دادن به وسیله ی فیلتر uppercase
۴۹.....	نشان دادن اندیس جدول (\$index)
۵۰.....	استفاده از \$seven و \$odd
۵۰.....	AngularJS و Twitter Bootstrap
۵۱.....	BootStrap
۵۱.....	کد HTML
۵۲.....	شرح دستورات بکار رفته در نمونه ی فوق
۵۳.....	تشریح کلاس های bootstrap
۵۴.....	کد جاوا اسکریپت
۵۵.....	شرح کد جاوا اسکریپت

۵۶.....	AngularJS – واکنشی و خواندن اطلاعات از پایگاه داده SQL
۵۸	نمونه هایی از کد سرور
۵۸	درخواست های HTTP از چندین سایت/cross-site
۵۸	کد سرور PHP و MySQL
۵۹.....	۲. کد سرور PHP و MS Access
۵۹.....	۳. کد سرور ASP.NET، VB Razor و SQL Lite
۶۰	۴. کد سرور ASP.NET، VB Razor و SQL Lite
۶۰	مدل شی گرای سند – AngularJS در HTML DOM
۶۱.....	دستور ng-disabled
۶۲	دستور ng-show
۶۲	دستور ng-hide
۶۳.....	رخدادها در angularJS
۶۳.....	دستور ng-click
۶۳.....	پنهان سازی المان های HTML
۶۴	نمایش دادن عناصر HTML
۶۵.....	فرم ها در AngularJS
۶۵.....	کنترل های HTML
۶۶	فرم های HTML
۶۶	نمونه ای از فرم AngularJS
۶۷.....	اعتبارسنجی ورودی در AngularJS
۶۷.....	Input Validation
۶۸.....	کد برنامه ی نمونه
۶۹	رابط برنامه سازی کاربردی (API) در AngularJS
۶۹	API ی سراسری AngularJS
۷۰.....	نمونه ای از کاربرد تابع angular.lowercase()
۷۰.....	نمونه ای از تابع angular.uppercase()
۷۱	نمونه ای از تابع angular.isString()
۷۱	تابع angular.isNumber()

۷۲.....	W3.CSS و AngularJS
۷۲.....	W3.CSS
۷۲.....	کد HTML
۷۳.....	توضیح Directive های استفاده شده در مثال بالا
۷۴.....	توصیف کلاس های W3.CSS
۷۴.....	کد JavaScript
۷۵.....	توضیح کدهای Java Script
۷۶.....	Includes در AngularJS
۷۶.....	تزریق کد AngularJS به همراه HTML
۷۷.....	انیمیشن AngularJS
۷۷.....	انیمیشن چیست؟
۷۸.....	به چه چیزهایی نیاز داریم؟
۷۸.....	ngAnimate چه کاری انجام می دهد؟
۷۹.....	انیمیشن به استفاده از CSS
۷۹.....	خاصیت انتقال (Transition) CSS ها
۷۹.....	خاصیت CSS Animation ها
۷۹.....	برنامه ی تحت وب AngularJS
۸۰.....	نمونه ای از یک برنامه ی AngularJS
۸۱.....	اسکلت و ساختار برنامه ی کاربردی AngularJS
۸۲.....	چکیده - برنامه چگونه کار می کند؟

مقدمه

زکات علم نشر آن است. حضرت علی(ع)

موسسه آموزشی تحلیل داده ، با حضور جمعی از متخصصین مجرب در زمینه برنامه نویسی در نظر دارد،مطالب آموزشی خود را در قالب کتاب های آموزشی و فیلم ، به صورت رایگان در دسترس عموم قرار دهد تا حتی آن دسته از عزیزانی که بنا به دلایل مالی،مسافت جغرافیایی و یا نداشتن وقت کافی ، امکان شرکت در دوره های حضوری برای آنها میسر نیست،از یادگیری بی بهره نمانند.

علاوه بر این علاقه مندان می توانند ، با ثبت نام در انجمن سایت تحلیل داده،سوالات خود را مطرح نموده و مدرسین آموزشگاه و اعضای انجمن در اسرع وقت،پاسخ های خود را، حتی الامکان به صورت فیلم، در دسترس عموم قرار دهند.

لذا از کلیه فعالان در این زمینه دعوت می شود، در این حرکت جمعی در کنار ما باشند و با حضور فعال خود در انجمن،گام موثری در بهبود سطح علمی جوانان کشور عزیزمان،ایران بردارند.

توجه :

برای دانلود سورس کد مثال های کتاب ،در بخش مقالات سایت به آموزش SPA در آدرس

www.tahlildadeh.com مراجعه فرمایید.

آموزش کتابخانه ی AngularJS

AngularJS یک کتابخانه ی جاوا اسکریپت است که قابلیت های جدید به HTML اضافه می کند.

AngularJS برای کار با SPA بسیار مناسب می باشد.

یادگیری آن بسیار آسان است.

مقاله ی حاضر برای آموزش سریع و کاربردی AngularJS تنظیم و طراحی شده است.

در ابتدا به آموزش اصول و مبانی این framework همانند: directive ها، expression ها، filter ها،

module ها و سرانجام controller ها می پردازیم. سپس به شرح مسائل پیچیده تر که برای کار با

AngularJS به آن ها نیاز دارید خواهیم پرداخت از جمله: Event ها، Form ها، Input، Validation، http و

غیره

نمونه ای از کاربرد AngularJS:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="">
    <p>Input something in the input box:</p>
    <p>Name : <input type="text" ng-model="name" placeholder="Enter name here"></p>
    <h1>Hello {{name}}</h1>
  </div>
</body>
</html>
```

آنچه که باید از قبل بدانید:

قبل از اینکه شروع به یادگیری کتابخانه ی AngularJS نمایید شما بایستی یک دانش ابتدایی از موارد زیر

داشته باشید:

HTML

CSS

JavaScript

پیشینه ی چارچوب کاری AngularJS

ویرایش ۱/۰ کتابخانه ی حاضر برای اولین بار در سال ۲۰۱۲ منتشر شد.

در سال ۲۰۰۹ یکی از کارمندان گوگل "Miško Hevery" کار بر روی کتابخانه ی مزبور را آغاز کرد.

این فکر با موفقیت به بار نشست و سرانجام پروژه بطور رسمی توسط شرکت گوگل پشتیبانی می شود.

مقدمه ای بر AngularJS

همان طور که پیشتر گفته شد، AngularJS صرفا یک کتابخانه یا چارچوب کاری JavaScript است. از این رو می توان با استفاده از تگ `<script>` آن را به صفحات HTML افزود. AngularJS خصیصه های (attribute) HTML را از طریق directive ها (دستورها) بسط داده و با بهره گیری از عبارت ها (expression) داده ها را به HTML مقید می سازد (bind می کند).

AngularJS یک کتابخانه یا چارچوب کاری JavaScript است!

همان طور که می دانید AngularJS یک چارچوب کاری (framework) است که با زبان جاوا اسکریپت نوشته شده است. AngularJS به صورت یک فایل جاوا اسکریپت ارائه شده و می توان آن را با استفاده از تگ `<script>` به صفحه ی وب اضافه کرد.

```
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
</script>
```

AngularJS قابلیت ها و امکانات زبان طراحی وب HTML را بسط می دهد!

کتابخانه ی AngularJS با استفاده از دستورهای ng-directive، قابلیت های HTML را بسط می دهد.

دستور ng-app یک برنامه ی AngularJS را تعریف یا مشخص می کند.

کار دستور **ng-model** این است که مقدار کنترل های **HTML** همچون **input** ، **select** و **textarea** را به داده های **app** متصل (**bind**) می کند.

دستور **ng-bind** داده های برنامه را به اشیا نمایی/دیداری (**view**) **HTML** مقید می سازد (**bind** می کند)؛ به عبارت دقیقتر به کمک آن داده ها نمایش داده می شوند.

مثالی از AngularJS

```
<div ng-app="">
  <p>Input something in the input box:</p>
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>
```

تشریح مثال:

AngularJS با بارگذاری کامل صفحه به صورت خودکار آغاز می شود.

پس از آن به واسطه ی دستور **ng-app** به **AngularJs** اطلاع دادیم که تگ **<div>** مالک برنامه ی **Angularjs** است.

سپس با استفاده از دستور **ng-model** مقدار فیلد **input** را به متغیر برنامه "name" مقید (**bind**) کردیم. در پایان دستور **ng-bind** کاری کردیم که متغیر برنامه ی **name** به **innerHTML** تگ **<p>** وصل (**bind**) شود.

AngularJS های Directive

همان طور که قبلا مشاهده کرده اید، **directive** های **AngularJS** در واقع همان خصیصه های (**attribute**) **HTML** هستند که دارای پیشوند **ng** می باشند.

به عنوان مثال دستور **ng-init**، متغیرهای برنامه را مقداردهی اولیه (**initialize**) می کند:

```
<div ng-app="" ng-init="firstName='John'">
  <p>The name is <span ng-bind="firstName"></span></p>
</div>
```

روشی جایگزین به همراه **HTML** معتبر:

```
<div data-ng-app="" data-ng-init="firstName='John'">
  <p>The name is <span data-ng-bind="firstName"></span></p>
</div>
```

چنانچه می خواهید کدهای **HTML** صفحه **Valid** یا معتبر باشند می توانید به جای پیشوند **ng-** از پیشوند **data-ng-** استفاده نمایید.

عبارت ها (expression) AngularJS

عبارت های **AngularJS** داخل **{{ }}** نوشته می شوند.

دستور **ng-init** متغیرهای برنامه ی **AngularJS** را مقداردهی اولیه می کند:

```
<div ng-app="" ng-init="firstName='John'">
  <p>The name is <span ng-bind="firstName"></span></p>
</div>
```

در کل، عبارت ها در **angularJs** اطلاعات را به المان های **HTML** مقید می سازند (منتقل می کنند) درست مشابه کاری که دستور **ng-bind** انجام می دهد.

```
<div ng-app="">
  <p>Input something in the input box:</p>
  <p>Name: <input type="text" ng-model="name"></p>
  <p>{{name}}</p>
</div>
```

برنامه های (application) AngularJS

ماژول های **Angularjs** برنامه های **Angularjs** را تعریف می کنند و کنترلر های **Angularjs** برنامه های **Angularjs** را مدیریت می کنند.

دستور **ng-app** اپلیکیشن مورد نظر را کنترل می کند و دستور **ng-controller** در حقیقت کنترلر را مشخص می کند.

```
<p>Try to change the names.</p>
<div ng-app="myApp" ng-controller="myCtrl">
  First Name: <input type="text" ng-model="firstName"><br>
  Last Name: <input type="text" ng-model="lastName"><br>
  <br>
  Full Name: {{firstName + " " + lastName}}
</div>
<script>
```

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function ($scope) {
  $scope.firstName = "John";
  $scope.lastName = "Doe";
});
</script>
```

ماژول های **AngularJS** اپلیکیشن را تعریف می کنند:

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

کنترلر (**controller**) اپلیکیشن ها را مدیریت می کنند:

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

عبارت ها/Expression در AngularJS

AngularJS با بهره گیری از عبارت ها داده ها را به **HTML** مقید (**bind**) می کند. عبارات در **AngularJS** داخل **{{ expression }}** نوشته و درج می شوند. کار آن ها این است که مثل دستور **ng-bind** داده ها رو به **HTML** مقید (**bind**) می کنند و در هر جایی که قرار دارند داده ها را جایگزین خود می سازند. عبارت های **AngularJS** بسیار شبیه به عبارت های جاوا اسکریپت هستند بدین معنا که می توانند مقادیر نوشتاری (**literal**)، عملگر (**operator**)، متغیر دربر داشته باشند.

مثال: **{{ 5 + 5 }}** یا **{{ firstName + " " + lastName }}**.

```
<div ng-app>
  <p>My first expression: {{ 5 + 5 }}</p>
</div>
```

حال اگر دستور **ng-app** را حذف کنید، **HTML** عبارت را بدون اینکه حل کند و همان گونه که هست نمایش می دهد:

```
<p>Without the ng-app directive, HTML will display the expression as it is, without solving it.</p>
<div>
  <p>My first expression: {{ 5 + 5 }}</p>
</div>
```

اعداد در AngularJS

اعداد **AngularJS** شبیه اعداد در زبان جاوا اسکریپت است:

```
<div ng-app="" ng-init="quantity=1;cost=5">
  <p>Total in dollar: {{ quantity * cost }}</p>
</div>
```

اکنون همین مثال را با استفاده از دستور **ng-bind** بکار می بریم:

```
<div ng-app="" ng-init="quantity=1;cost=5">
  <p>Total in dollar: <span ng-bind="quantity * cost"></span></p>
</div>
```

استفاده از دستور **ng-init** جهت مقداردهی اولیه ی متغیرها چندان رایج نیست. در مباحث بعدی به روش های دیگری برای مقداردهی اولیه خواهیم پرداخت.

رشته ها در AngularJS

رشته ها در **AngularJS** همان کاربردی را دارند که در جاوا اسکریپت دارند .

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">
  <p>The full name is: {{ firstName + " " + lastName }}</p>
</div>
```

اکنون همان مثال فوق را با استفاده از دستور **ng-bind** می نویسیم:

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">
  <p>The full name is: <span ng-bind="firstName + ' ' + lastName"></span></p>
</div>
```

اشیا در AngularJS

اشیا در جاوا اسکریپت مشابه اشیا در **AngularJS** هستند و روش ایجاد و بکارگیری آن ها نیز یکسان است:

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">
  <p>The name is {{ person.lastName }}</p>
</div>
```

حال همان مثال را در زیر مشاهده می کنید که با دستور **ng-bind** نوشته شده است:

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">
  <p>The name is <span ng-bind="person.lastName"></span></p>
</div>
```

آرایه ها در AngularJS

آرایه ها در جاوا اسکریپت و **AngularJS** یکی هستند:

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">
  <p>The third result is {{ points[2] }}</p>
</div>
```

حال همان مثال را با استفاده از دستور **ng-bind** می نویسیم:

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">
  <p>The third result is <span ng-bind="points[2]"></span></p>
</div>
```

مقایسه عبارات در JavaScript و Angularjs

درست مشابه عبارت ها در جاوااسکریپت، عبارت های **Angularjs** هم می تواند حامل مقادیر نوشتاری (**literal**) ، عملگرها (**operator** ها) و متغیرها باشد.

بر خلاف عبارت در جاوااسکریپت، عبارت ها **Angularjs** می توانند درون کدهای **HTML** درج گردند.

عبارت **Angularjs** از دستورات شرطی، حلقه ها و استثنا ها (**exception** ها) پشتیبانی نمی کند. بر خلاف آن عبارت در جاوااسکریپت از تمامی موارد نام برده پشتیبانی می کند.

عبارت ها در جاوااسکریپت قابلیت پشتیبانی از فیلترها را ندارند در حالی که عبارت های **Angularjs** از فیلترها پشتیبانی می کنند.

AngularJS در Modules

در **AngularJS**، ماژول یک برنامه را تعریف می کند.

ماژول در حقیقت نقش یک ظرف برای نگهداری بخش های مختلف یک برنامه را ایفا می کند.

همچنین می توان گفت که ماژول، یک ظرف برای کنترلگرهای برنامه است.

کنترلگرها همیشه متعلق به یک ماژول هستند.

یک ماژول به همراه یک کنترلگر

برنامه ی زیر ("myApp") دارای یک کنترلگر به نام ("myCtrl") است:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.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>
```

قرار دادن ماژول ها و کنترلگرها در فایل های جدا

در برنامه های **AngularJS**، متداول است که ماژول ها و کنترلگرها را در فایل های مجزای جاوا اسکریپت قرار می دهند.

در مثال زیر، فایل "**myApp.js**" دربردارنده ی یک ماژول برنامه است درحالی که فایل "**myCtrl.js**" حامل کنترلگر می باشد:

از پارامتر **[]** در تعریف ماژول می توان به منظور تعریف ماژول های وابسته بهره گرفت.

```
<!DOCTYPE html>
<html>
<head>
```

```

<title></title>
<script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="myCtrl">
    {{ firstName + " " + lastName }}
  </div>
  <script src="myApp.js"></script>
  <script src="myCtrl.js"></script>
</body>
</html>

```

توابع می توانند فضاهای نامی سراسری را تخریب کنند

در جاوااسکریپت باید از بکار بردن توابع سراسری تا حد امکان خودداری کرد، زیرا آنها می توانند به راحتی توسط دیگر اسکریپت ها بازنویسی یا نابود شوند.

ماژول های **AngularJS**، با قرار دادن توابع به صورت محلی درون ماژول مربوطه، این مشکل را تا حدی کاهش دهد.

زمان بارگذاری کتابخانه

اگرچه رایج است که در برنامه های **HTML**، اسکریپت ها در انتهای عنصر **<body>** قرار داده شوند، پیشنهاد می کنیم که شما فایل آدرس دهنده به کتابخانه ی **AngularJS** را در تگ **<head>** و یا در ابتدای المان **<body>** قرار دهید.

دلیل آن این است که فراخوانی **angular.module** تنها می تواند پس از بارگذاری کتابخانه، ترجمه (کامپایل) شود.

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.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>

```

Directives ها در AngularJS

Angular یک چارچوب کاری یا کتابخانه ی جاوا اسکریپت هست که به کمک خصیصه های نوینی به نام **Directive** ها (دستور ها)، به برنامه نویس یا طراح وب این اجازه را می دهد که **HTML** را بسط داده و قابلیت های آن را گسترش دهد.

Angular Directives

Directive ها با استفاده از پیشوند **ng**، خصیصه های **HTML** را بسط می دهند.

دستور **ng-app** یک برنامه ی **AngularJS** را مقدار دهی اولیه (آغاز و تعریف) می کند.

دستور **ng-init** داده های برنامه (**app data**) را مقدار دهی اولیه می کند.

دستور **ng-model** مقدار کنترل های دریافت ورودی **HTML** از قبیل **input** و **select** و **textarea** را به داده های برنامه مقید **bind** می کنند.

```

<!DOCTYPE html>
<html>
<head>
    <title></title>
    <script src="Angular.js"></script>
</head>
<body>
    <div ng-app="" ng-init="firstName='John'">
        <p>Input something in the input box:</p>
        <p>Name: <input type="text" ng-model="firstName"></p>
        <p>You wrote: {{ firstName }}</p>
    </div>
</body>
</html>

```

دستور **ng-app** همچنین به **AngularJS** اطلاع می دهد که المان **<div>** صاحب برنامه ی **AngularJS** می باشد.

اتصال داده (Data Binding)

عبارت **{{ firstName }}** که در مثال بالا مشاهده کردید، یک عبارت اتصال داده **AngularJS** است.

اتصال داده یا **data binding** عبارت های **AngularJS** را با داده های **AngularJS** هماهنگ و منطبق (**synchronize**) می سازد.

به طور مثال، **{{ firstName }}** با **ng-model="firstName"** برابر می باشد.

در مثال زیر مشاهده می کنید که دو فیلد دریافت متن (**text field**) با دستور **ng-model** مساوی و برابر شده اند:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div data-ng-app="" data-ng-init="quantity=1;price=5">
    <h2>Cost Calculator</h2>
    Quantity: <input type="number" ng-model="quantity">
    Price: <input type="number" ng-model="price">
    <p><b>Total in dollar:</b> {{quantity * price}}</p>
  </div>
</body>
</html>
```

همان طور که در مبحث پیشین مطرح شد، استفاده از دستور **ng-init** برای مقداردهی زیاد توصیه نمی شود. در مباحث آینده با روش بهتری برای مقدار دهی اولیه ی آشنا خواهیم شد.

تکرار کردن المان های HTML

دستور **ng-repeat** به منظور تکرار کردن المان **HTML** بکار رفته و عملکردی مشابه حلقه را دارد:

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div data-ng-app="" data-ng-init="names=['Jani','Hege','Kai']">
    <p>Looping with ng-repeat:</p>
    <ul>
      <li data-ng-repeat="x in names">
        {{ x }}
      </li>
    </ul>
  </div>
</body>
</html>

```

در زیر از دستور **ng-repeat** بر روی یک آرایه از اشیا استفاده شده است:

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="" ng-init="names=[
    {name:'Jani',country:'Norway'},
    {name:'Hege',country:'Sweden'},
    {name:'Kai',country:'Denmark'}]">
    <p>Looping with objects:</p>
    <ul>
      <li ng-repeat="x in names">
        {{ x.name + ', ' + x.country }}
      </li>
    </ul>
  </div>
</body>
</html>

```

AngularJS برای آن دسته از برنامه های کاربردی که نیاز به پیاده کردن عملیات **Create**، **Update**،

Delete، **Read** (ایجاد، خواندن، آپدیت، حذف) بر روی پایگاه داده دارند بسیار مناسب می باشد.

کافی است فرض کنید که این اشیا سطرهایی از یک پایگاه داده می باشند.

دستور ng-app

این **directive** در واقع عنصر آغازین (**root element**) یک برنامه ی کاربردی را تعریف می کند.

ng-app directive زمانی که صفحه بارگذاری می شود، به صورت خودکار راه اندازی (**auto-bootstrap**) و مقدار دهی اولیه می شود.

در فواصل آینده یاد خواهید گرفت چگونه **ng-app** می تواند به منظور اتصال کد ماژول ها، دارای یک مقدار (مانند **ng-app="myModule"**) باشد.

دستور in-init

این دستور مقادیر اولیه ی برنامه را مشخص می کند.

استفاده از **in-init** برای مقداردهی اولیه چندان متداول نیست، اغلب از یک کنترلر یا ماژول برای **initialization** استفاده می کنیم.

در مباحث بعدی درک بهتری نسبت به کنترلر ها و ماژول ها بدست خواهید آورد.

دستور ng-model

دستور **ng-model** مقادیر کنترل های دریافت کننده ورودی **HTML** همچون **input**، **select** و **textarea** را به داده های برنامه ی کاربردی متصل (**bind**) می کند.

دستور **ng-model** همچنین قادر به انجام عملیات زیر می باشد:

اعتبار سنجی داده های برنامه ی کاربردی از جمله عدد، ایمیل و اطلاعات لازم را بر عهده می گیرد.

وضعیت داده های برنامه را مشخص می کند: نامعتبر، **dirty** (دستخوش تغییر قرار گرفته)، خطا و دستکاری شده (**touched**) و غیره ...

کلاس های **css** را برای المان های **HTML** فراهم می نماید.

المان های **HTML** را به فرم های **HTML** مقید (**bind**) می کند.

دستور ng-repeat

دستور **ng-repeat** به ازای هر آیتم موجود در یک مجموعه یا آرایه، عناصر **HTML** را **clone** (تکثیر) می کند.

دایرکتیو ng-model در AngularJS

دایرکتیو **ng-model** مقادیر کنترلی **HTML** (**input**, **select**, **textarea**) را به داده های برنامه پیوند می دهد.

دایرکتیو ng-model

شما با استفاده از دایرکتیو **ng-model** می توانید مقدار یک **input** را به متغیری که در **AngularJS** ساخته شده است، پیوند دهید.

```
<div ng-app="myApp" ng-controller="myCtrl">
  Name: <input ng-model="name">
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope) {
    $scope.name = "John Doe";
  });
</script>
<p>Use the ng-model directive to bind the value of the input field to a property made in the controller.</p>
```

پیوند دو طرفه

پیوند می تواند دو طرفه باشد یعنی اگر کاربر مقدار ورودی را در **input** تغییر دهد، مقدار نظیر آن در **AngularJS** نیز تغییر کند:

```
<div ng-app="myApp" ng-controller="myCtrl">
  Name: <input ng-model="name">
  <h1>You entered: {{name}}</h1>
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope) {
    $scope.name = "John Doe";
  });
</script>
```

<p>Change the name inside the input field, and you will see the name in the header changes accordingly.</p>

اعتبارسنجی مقدار input کاربر

دایرکتیو **ng-model** می تواند مقدار ورودی یک برنامه را اعتبارسنجی کند. به عنوان مثال مقدار ورودی یک **input** را صرفاً از نوع **number** یا **email** قرار دهد و یا مقدار ورودی را الزامی (**required**) کند.

```
<form ng-app="" name="myForm">
  Email:
  <input type="email" name="myAddress" ng-model="text">
  <span ng-show="myForm.myAddress.$error.email">Not a valid e-mail address</span>
</form>
<p>Enter your e-mail address in the input field. AngularJS will display an error message if the address is not an e-mail.</p>
```

در مثال بالا، اگر عبارت داخل اتریبیوت **ng-show** مقدار **true** برگرداند، **span** نمایش داده خواهد شد.

نکته: اگر اتریبیوت **ng-model** وجود نداشته باشد، **AngularJS** به صورت پیش فرض یک **ng-model** برای خود می سازد.

وضعیت یک برنامه کاربردی (Application Status)

دایرکتیو **ng-model** می تواند وضعیت داده های یک برنامه را به صورت **invalid**، **dirty**، **touched** یا **error** مشخص کند.

```
<form ng-app="" name="myForm" ng-init="myText = 'post@myweb.com'">
  Email:
  <input type="email" name="myAddress" ng-model="myText" required>
  <p>Edit the e-mail address, and try to change the status.</p>
  <h1>Status</h1>
  <p>Valid: {{myForm.myAddress.$valid}} (if true, the value meets all criteria).</p>
  <p>Dirty: {{myForm.myAddress.$dirty}} (if true, the value has been changed).</p>
  <p>Touched: {{myForm.myAddress.$touched}} (if true, the field has been in focus).</p>
</form>
```

کلاس های CSS

دایرکتیو **ng-model** با در نظر گرفتن وضعیت المان های **HTML** می تواند خاصیت کلاس های **Css** را به آنها اضافه کند.

```
<form ng-app="" name="myForm">
  Enter your name:
  <input name="myName" ng-model="myText" required>
</form>
<p>Edit the text field and it will get/lose classes according to the status.</p>
<p><b>Note:</b> A text field with the "required" attribute is not valid when it is empty.</p>
```


دایرکتیو **ng-model** با توجه با وضعیت فیلد های فرم می تواند کلاس های زیر را اضافه و یا حذف کند.

- **ng-empty**
- **ng-not-empty**
- **ng-touched**
- **ng-untouched**
- **ng-valid**
- **ng-invalid**
- **ng-dirty**
- **ng-pending**
- **ng-pristine**

لیست انتخاب (Dropdown) در AngularJS

AngularJS به شما این امکان را می دهد تا بر اساس لیستی از عناصر درون یک آرایه و یا یک شی (object)، یک dropdown بسازید.

ساختن یک dropdown با استفاده از ng-options

اگر می خواهید با استفاده از یک **object** یا یک آرایه در **AngularJS** یک **dropdown** بسازید باید از دایرکتیو **ng-option** استفاده کنید. به عنوان مثال:

```
<div ng-app="myApp" ng-controller="myCtrl">
  <select ng-model="selectedName" ng-options="x for x in names"></select>
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope) {
    $scope.names = ["Emil", "Tobias", "Linus"];
  });
</script>
<p>This example shows how to fill a dropdown list using the ng-options directive.</p>
```

مقایسه ng-repeat با ng-options

شما همچنین می توانید از دایرکتیو **ng-repeat** برای ساخت یک **dropdown** مشابه استفاده کنید:

```
<div ng-app="myApp" ng-controller="myCtrl">
  <select>
    <option>کنید انتخاب</option>
    <option ng-repeat="x in names">{{x}}</option>
  </select>
```

```

</select>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function ($scope) {
    $scope.names = ["Emil", "Tobias", "Linus"];
});
</script>
<p>This example shows how to fill a dropdown list using the ng-repeat directive.</p>

```

دایرکتیو **ng-repeat** یک بلاک از کد **HTML** را برای هر عنصر داخل آرایه تکرار می کند و به همین دلیل می توان از آن برای ساخت گزینه های یک **dropdown** استفاده کرد، اما دایرکتیو **ng-options** به صورت اختصاصی برای ساختن گزینه های یک **dropdown** ساخته شده است و همچنین یک مزیت بزرگ نسبت به **ng-repeat** دارد. **Dropdown** هایی که با **ng-options** ساخته می شوند، گزینه های انتخاب شده را یک **object** و همچنین **dropdown** هایی که با **ng-repeat** ساخته می شوند گزینه های انتخاب شده را یک رشته در نظر می گیرند.

برای چه باید استفاده کنیم؟

فرض کنید مانند شکل زیر یک آرایه ای از **object** ها به صورت زیر در اختیار داریم:

```

$scope.cars = [
    { model: "Ford Mustang", color: "red" },
    { model: "Fiat 500", color: "white" },
    { model: "Volvo XC90", color: "black" }
];

```

دایرکتیو **ng-repeat** محدودیت هایی دارد و هر گزینه انتخاب شده را یک رشته در نظر می گیرد:

```

<div ng-app="myApp" ng-controller="myCtrl">
    <p>Select a car:</p>
    <select ng-model="selectedCar">
        <option ng-repeat="x in cars" value="{{x.model}}">{{x.model}}</option>
    </select>
    <h1>You selected: {{selectedCar}}</h1>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function ($scope) {
    $scope.cars = [
        { model: "Ford Mustang", color: "red" },
        { model: "Fiat 500", color: "white" },
        { model: "Volvo XC90", color: "black" }
    ];
});

```

```

    ];
  });
</script>
<p>When you use the ng-repeat directive to create dropdown lists, the selected value must be a string.</p>
<p>In this example you will have to choose between the color or the model to be your selected value.</p>

```

زمانی که از دایرکتیو **ng-options** استفاده می کنید، هر گزینه انتخاب شده را یک **object** در نظر می گیرد:

```

<div ng-app="myApp" ng-controller="myCtrl">
  <p>Select a car:</p>
  <select ng-model="selectedCar" ng-options="x.model for x in cars"></select>
  <h1>You selected: {{selectedCar.model}}</h1>
  <p>It's color is: {{selectedCar.color}}</p>
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope) {
    $scope.cars = [
      { model: "Ford Mustang", color: "red" },
      { model: "Fiat 500", color: "white" },
      { model: "Volvo XC90", color: "black" }
    ];
  });
</script>
<p>When you use the ng-options directive to create dropdown lists, the selected value can be an object.</p>
<p>In this example you can display both the model and the color of the selected element.</p>

```

گزینه های انتخاب شده به صورت **object**، اطلاعات بیشتری درباره ی آن گزینه در خود نگه داری می کند و برنامه ی شما را انعطاف پذیرتر خواهد کرد. در این مقاله ما از دایرکتیو **ng-options** استفاده می کنیم.

منبع داده (Data Source) به صورت یک object

در مثال هایی که در بالا گفته شد، **data source** ما یک آرایه بود اما ما می توانیم از **object** هم به عنوان **data source** استفاده کنیم.

فرض کنید یک **object** جفت مقدار ارزشی (**key-value pairs**) در اختیار داریم:

```

<div ng-app="myApp" ng-controller="myCtrl">
  <p>Select a car:</p>
  <select ng-model="selectedCar" ng-options="x for (x, y) in cars"></select>
  <h1>You selected: {{selectedCar}}</h1>
</div>
<p>This example demonstrates the use of an object as the data source when creating a dropdown list.</p>

```

```
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function ($scope) {
  $scope.cars = {
    car01: "Ford",
    car02: "Fiat",
    car03: "Volvo"
  }
});
</script>
```

بیان اتریوت **ng-options** برای **object** ها مقداری متفاوت است. در مثال زیر، از یک **object** به عنوان **data source** استفاده شده است و **x** بیانگر کلید (**key**) و **y** (**value**) بیانگر مقدار می باشد:

```
<div ng-app="myApp" ng-controller="myCtrl">
  <p>Select a car:</p>
  <select ng-model="selectedCar" ng-options="x for (x, y) in cars"></select>
  <h1>You selected: {{selectedCar.brand}}</h1>
  <h2>Model: {{selectedCar.model}}</h2>
  <h3>Color: {{selectedCar.color}}</h3>
  <p>Note that the selected value represents an object.</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function ($scope) {
  $scope.cars = {
    car01: { brand: "Ford", model: "Mustang", color: "red" },
    car02: { brand: "Fiat", model: "500", color: "white" },
    car03: { brand: "Volvo", model: "XC90", color: "black" }
  }
});
</script>
```

در **key-value pair** ، گزینه انتخاب شده همیشه **value** می باشد. **value** در **key-value pair** می تواند یک **object** باشد. در مثال زیر خواهید دید که مقدار انتخاب شده در **key-value pair** ، **value** و فقط یک مرتبه به صورت **object** خواهد بود.

```
<div ng-app="myApp" ng-controller="myCtrl">
  <p>Select a car:</p>
  <select ng-model="selectedCar" ng-options="y.brand for (x, y) in cars"></select>
  <h1>You selected: {{selectedCar.brand}}</h1>
  <h2>Model: {{selectedCar.model}}</h2>
  <h3>Color: {{selectedCar.color}}</h3>
  <p>The visible text inside the dropdown list can also be a property of the value object.</p>
```

```

</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function ($scope) {
    $scope.cars = {
        car01: { brand: "Ford", model: "Mustang", color: "red" },
        car02: { brand: "Fiat", model: "500", color: "white" },
        car03: { brand: "Volvo", model: "XC90", color: "black" }
    }
});
</script>

```

Controller ها در AngularJS

کنترلرها در واقع داده های برنامه را مدیریت یا کنترل می کنند.

کنترلرهای **AngularJS** همان اشیای منظم و رایج **JavaScript** می باشند.

controller ها کلاسهای جاوا اسکریپت هستند. زمانی که کاربر با برنامه تعامل برقرار می کند **controller** پاسخگوی مستقیم دستورات کاربر می باشد و مدل را تغییر می دهد. به کنترلرها بخش منطق برنامه نیز گفته می شود.

AngularJS Controllers

برنامه های **AngularJS** توسط کنترلرها مدیریت می شوند.

دستور **ng-controller** در حقیقت کنترلگر برنامه ی مورد نظر را تعریف می کند.

یک کنترلر، در واقع یک شی جاوا اسکریپت است که به واسطه ی یک سازنده ی شی (**object constructor**) ساخته شده است.

```

<div ng-app="myApp" ng-controller="myCtrl">
    First Name: <input type="text" ng-model="firstName"><br>
    Last Name: <input type="text" ng-model="lastName"><br>
    Full Name: {{firstName + " " + lastName}}
</div>
<script>
var app = angular.module('myApp', []);

```

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

تشریح برنامه ی فوق

برنامه ی مورد نظر به وسیله ی دستور یا **ng-app="myApp" directive** اعلان شده است. این برنامه داخل عنصر **<div>** کار می کند.

خیمیه ی **"ng-controller="myCtrl"** یک **directive** (دستور) **AngularJS** می باشد و یک کنترلگر را مشخص می کند.

تابع **myCtrl** یک تابع جاوا اسکریپت است.

AngularJS به وسیله ی شی **\$scope** کنترلگر مورد نظر را فرامی خواند.

\$scope در **AngularJS**، همان شی برنامه ی مورد نظر می باشد که حاوی متغیرها و توابع برنامه می باشد.

کنترلگر مورد نظر، دو خاصیت (متغیر) را داخل شی **scope** ایجاد می کند (متغیرهای **firstName** و **lastName**).

دستور **ng-model** فیلدهای دریافت کننده ی ورودی را به خاصیت های کنترل مورد نظر مقید/ **bind** می کند (**firstName** و **lastName**).

متدهای کنترلگر

نمونه ی بالا یک شی کنترلگر را به همراه دو خاصیت نمایش می دهد: **firstName** و **lastName**.

یک کنترلگر همچنین می تواند دارای توابع یا متدهایی باشد (متغیرهایی به عنوان تابع):

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
```

```

</head>
<body>
  <div ng-app="myApp" ng-controller="personCtrl">
    First Name: <input type="text" ng-model="firstName"><br>
    Last Name: <input type="text" ng-model="lastName"><br>
    <br>
    Full Name: {{fullName()}}
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('personCtrl', function ($scope) {
      $scope.firstName = "John";
      $scope.lastName = "Doe";
      $scope.fullName = function () {
        return $scope.firstName + " " + $scope.lastName;
      }
    });
  </script>
</body>
</html>

```

تعریف کنترلرها در فایل های خارجی

در برنامه های بزرگ، اغلب کنترلرها را درون فایل های خارجی ذخیره می کنند.

کافی است کد بین دو تگ باز و بسته ی **<script>** را درون فایل خارجی به نام **personController.js** جای گذاری کنید:

```

angular.module('myApp', []).controller('personCtrl', function ($scope) {
  $scope.firstName = "John",
  $scope.lastName = "Doe",
  $scope.fullName = function () {
    return $scope.firstName + " " + $scope.lastName;
  }
});

```

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="personCtrl">
    First Name: <input type="text" ng-model="firstName"><br>
    Last Name: <input type="text" ng-model="lastName"><br>

```

```

    <br>
    Full Name: {{firstName + " " + lastName}}
  </div>
  <script src="personController.js"></script>
</body>
</html>

```

مثال دیگر

برای مثال بعدی یک فایل کنترلر جدید ایجاد می کنیم:

```

angular.module('myApp', []).controller('namesCtrl', function ($scope) {
  $scope.names = [
    { name: 'Jani', country: 'Norway' },
    { name: 'Hege', country: 'Sweden' },
    { name: 'Kai', country: 'Denmark' }
  ];
});

```

فایل مورد نظر را به این نام ذخیره کنید: **namesController.js**:

حال فایل کنترلر را داخل یک برنامه بکار ببرید:

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="namesCtrl">
    <ul>
      <li ng-repeat="x in names">
        {{ x.name + ', ' + x.country }}
      </li>
    </ul>
  </div>
  <script src="namesController.js"></script>
</body>
</html>

```

محدوده (Scope) در AngularJS

Scope را می توان به چند طریق تعریف کرد:

به قسمتی که HTML (view) و JavaScript (Controller) را بهم پیوند می دهد ، Scope می گویند.

به یک object با ویژگی ها و متدهای قابل دسترس نیز Scope می گویند. Scope هم برای view و هم برای Controller قابل دسترس می باشد.

چگونه از Scope استفاده کنیم؟

زمانی که شما یک Controller در AngularJS ایجاد می کنید، یک \$scope object به عنوان argument ایجاد می شود. در مثال زیر، ویژگی های ساخته شده در Controller، می توانند به view ارجاع داده شوند.

```
<div ng-app="myApp" ng-controller="myCtrl">
  <h1>{{carname}}</h1>
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope) {
    $scope.carname = "Volvo";
  });
</script>
<p>The property "carname" was made in the controller, and can be referred to in the view by using the {{ }}
brackets.</p>
```

هرگاه به \$scope در Controller ویژگی هایی اضافه کنیم، view (HTML) می تواند به این ویژگی ها دسترسی داشته باشد. شما نباید از پیشوند \$scope در view استفاده کنید بلکه فقط باید به نام آن ویژگی ارجاع دهید، به عنوان مثال باید به صورت {{carname}} بنویسید.

درک کردن Scope

یک برنامه کاربردی AngularJS شامل بخش های زیر می باشد:

- ۴ View که شامل کدهای Html است.
- ۴ Model که view مربوط به آن دسترسی دارد.
- ۴ Controller که با استفاده از توابع JavaScript، اطلاعات را ایجاد کند، تغییر دهد، حذف کند و یا کنترل کند.

Scope یک object با ویژگی ها و متدهایی است که هم برای view و Controller قابل دسترسی باشد. به عنوان مثال، هر گاه شما در view تغییراتی ایجاد کنید، model و Controller به روز رسانی می شوند.

```
<div ng-app="myApp" ng-controller="myCtrl">
  <input ng-model="name">
  <h1>My name is {{name}}</h1>
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope) {
    $scope.name = "John Doe";
```

```
});
</script>
<p>When you change the name in the input field, the changes will affect the model, and it will also affect the name property in the controller.</p>
```

Scope خود را بشناسید

این نکته بسیار مهم است که شما بدانید از کدام **Scope** باید استفاده کنید. در دو مثال قبلی تنها یک **Scope** وجود داشت و شناختن **Scope** کار دشواری نبود اما برای برنامه های بزرگتر **Section** هایی از **HTML DOM** وجود دارد که برای هر **Scope** یکتای خود قابل دسترسی می باشد.

به عنوان مثال، هرگاه از دایرکتیو **ng-repeat** استفاده می کنیم، هر تکرار به همان **object** تکرار شده دسترسی دارد.

```
<div ng-app="myApp" ng-controller="myCtrl">
  <ul>
    <li ng-repeat="x in names">{{x}}</li>
  </ul>
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope) {
    $scope.names = ["Emil", "Tobias", "Linus"];
  });
</script>
<p>The variable "x" has a different value for each repetition, proving that each repetition has its own scope.</p>
```

هر المان به همان **object** تکرارشونده دسترسی دارد. در این مورد یک رشته با **x** نشان داده شده است.

Scope ریشه

هر برنامه ی کاربردی، یک **\$rootScope** دارد که **Scope** ساخته شده در المان **Html** آن دارای دایرکتیو **ng-app** می باشد. **rootScope** در کل برنامه قابل دسترسی است و اگر یک متغیر دارای نام های مشابه در **Current Scope** و **rootScope** باشد، برنامه یکی از آنها را استفاده می کند.

به عنوان مثال، یک متغیر به نام **color** هم در **Controller Scope** و هم در **rootScope** وجود دارد.

```
<body ng-app="myApp">
  <p>The rootScope's favorite color:</p>
  <h1>{{color}}</h1>
  <div ng-controller="myCtrl">
    <p>The scope of the controller's favorite color:</p>
    <h1>{{color}}</h1>
  </div>
  <p>The rootScope's favorite color is still:</p>
```

```

<h1>{{color}}</h1>
<script>
var app = angular.module('myApp', []);
app.run(function ($rootScope) {
    $rootScope.color = 'blue';
});
app.controller('myCtrl', function ($scope) {
    $scope.color = "red";
});
</script>
<p>Notice that controller's color variable does not overwrite the rootScope's color value.</p>
</body>

```

فیلترها در AngularJS

فیلترها را می توان با یک کاراکتر " | " به عبارات (**expression**) و **directive** (دستورات) اضافه کرد.

با بهره گیری از فیلترهای **AngularJS** می توان داده ها را تبدیل کرد:

فیلتر	شرح
currency	عدد را با فرمت پول رایج نمایش می دهد/فرمت عدد به پول رایج.
filter	یک زیر مجموعه از آیتم های موجود در یک آرایه را انتخاب کرده و برمی گرداند.
lowercase	فرمت یک رشته را به حروف کوچک تبدیل می کند و آن را با حروف کوچک نمایش می دهد.
orderBy	آرایه را بر اساس یک رشته مرتب می سازد.
uppercase	فرمت رشته را به حروف بزرگ تبدیل می کند.

افزودن فیلتر به عبارت ها

همان طور که بالا نیز گفته شد، یک فیلتر را می توان به وسیله ی کاراکتر " | " به عبارت مد نظر اضافه کرد. در

دو مثال بعدی از همان کنترلر **person** درس پیشین استفاده می کنیم.

فیلتر **uppercase** رشته ها را با حروف بزرگ نمایش می دهد:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="personCtrl">
    <p>The name is {{ lastName | uppercase }}</p>
  </div>
  <script src="personController.js"></script>
</body>
</html>
```

فیلتر **lowercase** رشته ها را با حروف کوچک تبدیل کرده و نمایش می دهد:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="personCtrl">
    <p>The name is {{ lastName | lowercase }}</p>
  </div>
  <script src="personController.js"></script>
</body>
</html>
```

فیلتر currency

فرمت یک عدد را به فرمت پول رایج تبدیل کرده یا به عبارتی یک عدد را به صورت (در قالب) پول رایج نمایش

می دهد:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
```

```

</head>
<body>
  <div ng-app="myApp" ng-controller="costCtrl">
    Quantity: <input type="number" ng-model="quantity">
    Price: <input type="number" ng-model="price">
    <p>Total = {{ (quantity * price) | currency }}</p>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('costCtrl', function ($scope) {
      $scope.quantity = 1;
      $scope.price = 9.99;
    });
  </script>
</body>
</html>

```

افزودن فیلتر به Directive ها

فیلتر را می توان با استفاده از کاراکتر " | " به **directive** مورد نظر ضمیمه کرد.

یک آرایه را به وسیله ی یک عبارت (**expression**) مرتب می سازد:

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="namesCtrl">
    <p>Looping with objects:</p>
    <ul>
      <li ng-repeat="x in names | orderBy:'country'">
        {{ x.name + ', ' + x.country }}
      </li>
    </ul>
  </div>
  <script src="namesController.js"></script>
</body>
</html>

```

فیلتر کردن ورودی

یک **filter** که برای فیلتر کردن و انتخاب ورودی های خاص در نظر گرفته شده است، به واسطه ی یک کاراکتر

" | " به **directive** پیوست شده و به دنبال آن فیلتر مورد نظر، کاراکتر دو نقطه " : " و نام مدل تایپ می شوند.

Filter یک تعداد مشخص یا زیرمجموعه ای از آیتم های آرایه را انتخاب کرده و برمی گرداند:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="namesCtrl">
    <p>Filtering input:</p>
    <p><input type="text" ng-model="test"></p>
    <ul>
      <li ng-repeat="x in names | filter:test | orderBy:'country'">
        {{ (x.name | uppercase) + ', ' + x.country }}
      </li>
    </ul>
  </div>
  <script src="namesController.js"></script>
</body>
</html>
```

سرویس های AngularJS

شما در **AngularJS** می توانید سرویس شخصی برای خودتان بسازید و یا از سرویس های پیش ساخته زیادی که در اختیار دارید، استفاده کنید.

سرویس چیست؟

در **AngularJS**، سرویس یک تابع و یا **object** است که برای برنامه کاربردی **AngularJS** قابل دسترس و یا محدود شده باشد. **AngularJS** حدود ۳۰ سرویس پیش ساخته شده دارد که یکی از آنها، سرویس **\$location** می باشد. سرویس **\$location** دارای متدهایی است که اطلاعات موقعیت مکانی را به صفحه وب (**Web page**) ارسال می کند. در مثال زیر از **\$location** به عنوان یک **Controller** استفاده شده است:

```
<div ng-app="myApp" ng-controller="myCtrl">
  <p>The url of this page is:</p>
  <h3>{{myUrl}}</h3>
</div>
<p>This example uses the built-in $location service to get the absolute url of the page.</p>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope, $location) {
```

```

$scope.myUrl = $location.absUrl();
});
</script>

```

به خاطر داشته باشید که سرویس `$location` به عنوان یک `argument` در `Controller` تعریف می شود. به منظور استفاده از سرویس در `Controller` باید آن را به صورت وابسته تعریف کرد.

چرا از سرویس ها استفاده می کنیم؟

سرویس `$location` را در نظر بگیرید. به نظر می رسد که به جای آن می توان از `object` هایی که در `DOM` هستند مانند `window.location` استفاده کرد - البته می توان این کار را انجام داد - اما در برنامه `AngularJS`، شما محدودیت هایی خواهید داشت. `AngularJS` دائما برنامه شما را نظارت می کند و برای مدیریت تغییرات و ویژگی های رخدادها، ترجیح می دهد شما از `$location` به جای `window.location` استفاده کنید.

سرویس \$http

سرویس `$http` یکی از سرویس های پرکاربرد در برنامه های `AngularJS` می باشد. این سرویس یک درخواست به سرور ارسال می کند و به شما اجازه می دهد پاسخ برگشتی را مدیریت کنید. در مثال بسیار ساده زیر از سرویس `$http` برای ارسال درخواست به سرور استفاده شده است.

```

<div ng-app="myApp" ng-controller="myCtrl">
  <p>Today's welcome message is:</p>
  <h1>{{myWelcome}}</h1>
</div>
<p>The $http service requests a page on the server, and the response is set as the value of the "myWelcome"
variable.</p>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope, $http) {
    $http.get("welcome.htm").then(function (response) {
      $scope.myWelcome = response.data;
    });
  });
</script>

```

سرویس \$timeout

سرویس `$timeout` نمونه `Angular` تابع `window.setTimeout` می باشد. در مثال زیر بعد از دو ثانیه، یک پیغام جدید به نمایش در خواهد آمد.

```

<div ng-app="myApp" ng-controller="myCtrl">
  <p>This header will change after two seconds:</p>
  <h1>{{myHeader}}</h1>

```

```

</div>
<p>The $timeout service runs a function after a sepecified number of milliseconds.</p>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope, $timeout) {
    $scope.myHeader = "Hello World!";
    $timeout(function () {
      $scope.myHeader = "How are you today?";
    }, 2000);
  });
</script>

```

سرویس \$interval

سرویس **\$interval** نمونه **Angular** تابع **window.setInterval** می باشد. در مثال زیر در هر ثانیه، زمان را نمایش خواهد داد.

```

<div ng-app="myApp" ng-controller="myCtrl">
  <p>The time is:</p>
  <h1>{{theTime}}</h1>
</div>
<p>The $interval service runs a function every specified millisecond.</p>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope, $interval) {
    $scope.theTime = new Date().toLocaleTimeString();
    $interval(function () {
      $scope.theTime = new Date().toLocaleTimeString();
    }, 1000);
  });
</script>

```

ساختن سرویس شخصی

برای ساختن یک سرویس شخصی، سرویس خود را به ماژول متصل کنید. در مثال زیر یک سرویس به نام **hexafy** می سازیم:

```

app.service('hexafy', function () {
  this.myFunc = function (x) {
    return x.toString(16);
  }
});

```

زمانی که یک فیلتر تعریف می کنید، برای استفاده از سرویس شخصی ساخته شده ی خود، آن را به صورت وابسته اضافه کنید. در مثال زیر، برای تبدیل عدد صحیح به عدد هگزا دسیمال، از سرویس **hexafy** که قبلا ساختیم استفاده کرده ایم.


```

<div ng-app="myApp" ng-controller="myCtrl">
  <p>The hexadecimal value of 255 is:</p>
  <h1>{{hex}}</h1>
</div>
<p>A custom service whith a method that converts a given number into a hexadecimal number.</p>
<script>
  var app = angular.module('myApp', []);
  app.service('hexafy', function () {
    this.myFunc = function (x) {
      return x.toString(16);
    }
  });
  app.controller('myCtrl', function ($scope, hexafy) {
    $scope.hex = hexafy.myFunc(255);
  });
</script>

```

استفاده کردن از یک سرویس دلخواه داخل یک فیلتر

زمانی که شما یک سرویس را طراحی می کنید و آن را به برنامه خود متصل می کنید، می توانید از آن در هر **controller**، **directive**، **filter** و یا حتی داخل سرویس های دیگر استفاده کنید. برای استفاده از سرویس داخل یک فیلتر، بعد از تعریف کردن فیلتر، سرویس را به صورت وابسته به آن اضافه کنید. در مثال زیر از سرویس **hexafy** داخل فیلتر **myFormat** استفاده شده است.

```

<div ng-app="myApp">
  Convert the number 255, using a custom made service inside a custom made filter:
  <h1>{{255 | myFormat}}</h1>
</div>
<script>
  var app = angular.module('myApp', []);
  app.service('hexafy', function () {
    this.myFunc = function (x) {
      return x.toString(16);
    }
  });
  app.filter('myFormat', ['hexafy', function (hexafy) {
    return function (x) {
      return hexafy.myFunc(x);
    };
  }]);
</script>

```

شما می توانید از فیلتر در هنگام نمایش دادن مقادیر از یک **object** و یا یک آرایه استفاده کنید.

```

<div ng-app="myApp" ng-controller="myCtrl">
  <p>Use a filter when displayin the array [255, 251, 200]:</p>
  <ul>
    <li ng-repeat="x in counts">{{x | myFormat}}</li>
  </ul>

```

```

</ul>
<p>This filter uses a service that converts numbers into hexadecimal values.</p>
</div>
<script>
var app = angular.module('myApp', []);
app.service('hexafy', function () {
    this.myFunc = function (x) {
        return x.toString(16);
    }
});
app.filter('myFormat', ['hexafy', function (hexafy) {
    return function (x) {
        return hexafy.myFunc(x);
    };
}]);
app.controller('myCtrl', function ($scope) {
    $scope.counts = [255, 251, 200];
});
</script>

```

AngularJS و سرویس \$http

\$http یک سرویس AngularJS است که جهت خواندن داده ها از سرورهای راه دور (remote server) بکار می رود.

فراهم کردن دادن داده ها

داده ی زیر می تواند به وسیله ی یک سرویس دهنده ی وب (web server) ارائه شود:

```

{
  "records": [
    {
      "Name": "Alfreds Futterkiste",
      "City": "Berlin",
      "Country": "Germany"
    },
    {
      "Name": "Berglunds snabbköp",
      "City": "Luleå",
      "Country": "Sweden"
    },
    {
      "Name": "Centro comercial Moctezuma",
      "City": "México D.F.",

```

```
"Country" : "Mexico"
},
{
  "Name" : "Ernst Handel",
  "City" : "Graz",
  "Country" : "Austria"
},
{
  "Name" : "FISSA Fabrica Inter. Salchichas S.A.",
  "City" : "Madrid",
  "Country" : "Spain"
},
{
  "Name" : "Galería del gastrónomo",
  "City" : "Barcelona",
  "Country" : "Spain"
},
{
  "Name" : "Island Trading",
  "City" : "Cowes",
  "Country" : "UK"
},
{
  "Name" : "Königlich Essen",
  "City" : "Brandenburg",
  "Country" : "Germany"
},
{
  "Name" : "Laughing Bacchus Wine Cellars",
  "City" : "Vancouver",
  "Country" : "Canada"
},
{
  "Name" : "Magazzini Alimentari Riuniti",
  "City" : "Bergamo",
  "Country" : "Italy"
},
{
  "Name" : "North/South",
  "City" : "London",
  "Country" : "UK"
},
{
  "Name" : "Paris spécialités",
  "City" : "Paris",
  "Country" : "France"
},
{
  "Name" : "Rattlesnake Canyon Grocery",
  "City" : "Albuquerque",
```

```
    "Country" : "USA"
  },
  {
    "Name" : "Simons bistro",
    "City" : "København",
    "Country" : "Denmark"
  },
  {
    "Name" : "The Big Cheese",
    "City" : "Portland",
    "Country" : "USA"
  },
  {
    "Name" : "Vaffeljernet",
    "City" : "Århus",
    "Country" : "Denmark"
  },
  {
    "Name" : "Wolski Zajazd",
    "City" : "Warszawa",
    "Country" : "Poland"
  }
]
```

AngularJS \$http

یک سرویس اصلی است که برای خواندن اطلاعات از سرویس دهنده های وب کاربرد دارد.

`$http.get(url)` یک تابع است که برای خواندن داده های سرویس دهنده مورد استفاده قرار می گیرد.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="customersCtrl">
    <ul>
      <li ng-repeat="x in names">
        {{ x.Name + ', ' + x.Country }}
      </li>
    </ul>
  </div>
<script>
  var app = angular.module('myApp', []);
  app.controller('customersCtrl', function ($scope, $http) {
    $http.get("customers.htm")
```

```

        .then(function (response) { $scope.names = response.data.records; });
    });
</script>
</body>
</html>

```

شرح برنامه:

دستور **ng-app**، همان طور که در فوایل پیشین شرح داده شد برنامه ی **AngularJS** را تعریف کرده و به عبارتی عنصر آغازین را مشخص می کند. برنامه ی حاضر درون یک تگ **<div>** اجرا می شود.

ng-controller را می توان شی کنترلگر (**controller object**) نامید.

تابع **customersCtrl** یک سازنده ی شی (**object constructor**) متعارف جاوا اسکریپت است.

AngularJS با استفاده از اشیا **\$scope** و **\$http**، تابع **customersCtrl** را فرامی خواند.

\$scope در واقع شی **application** است (همان مالک و متغیرها و توابع برنامه).

سرویس **\$http** یک شی **XMLHttpRequest** است که توسط آن داده های خارجی را درخواست می کنید.

\$http.get() اطلاعات **json** را از آدرس **http://www.Tahlildadeh.com** می خواند.

در صورت موفقیت، کنترلگر مورد نظر یک خاصیت (**names**) را در شی **scope** با داده های **JSON** از سرویس دهنده، ایجاد می کند.

جداول در AngularJS

دستور **ng-repeat** بهترین گزینه برای نمایش دادن جدول ها می باشد.

نمایش دادن داده ها در جدول

نمایش دادن جدول ها توسط **AngularJS** امر بسیار ساده ای می باشد:

```
<!DOCTYPE html>
```

```
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <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("customers.html")
      .then(function (response) { $scope.names = response.data.records; });
  });
</script>
</body>
</html>
```

نمایش دادن اطلاعات به وسیله ی استایل CSS

به منظور بهتر کردن ظاهر آن می توان مقداری **CSS** به صفحه اضافه کرد:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
  <style>
    table, th, td {
      border: 1px solid grey;
      border-collapse: collapse;
      padding: 5px;
    }
    table tr:nth-child(odd) {
      background-color: #f1f1f1;
    }
    table tr:nth-child(even) {
      background-color: #ffffff;
    }
  </style>
</head>
<body>
```

```
<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("customers.html")
    .then(function (response) { $scope.names = response.data.records; });
});
</script>
</body>
</html>
```

به منظور مرتب سازی جدول مورد نظر، لازم است یک فیلتر **orderBy** اضافه کنید:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
  <style>
    table, th, td {
      border: 1px solid grey;
      border-collapse: collapse;
      padding: 5px;
    }
    table tr:nth-child(odd) {
      background-color: #f1f1f1;
    }

    table tr:nth-child(even) {
      background-color: #ffffff;
    }
  </style>
</head>
<body>
  <div ng-app="myApp" ng-controller="customersCtrl">
    <table>
      <tr ng-repeat="x in names | orderBy : 'Country'">
        <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("customers.html")
    .then(function (response) { $scope.names = response.data.records; });
});
</script>
</body>
</html>

```

نمایش دادن به وسیله ی فیلتر uppercase

جهت نمایش دادن با حروف بزرگ، کافی است فیلتر **uppercase** را اضافه کنید:

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
  <style>
    table, th, td {
      border: 1px solid grey;
      border-collapse: collapse;
      padding: 5px;
    }
    table tr:nth-child(odd) {
      background-color: #f1f1f1;
    }
    table tr:nth-child(even) {
      background-color: #ffffff;
    }
  </style>
</head>
<body>
  <div ng-app="myApp" ng-controller="customersCtrl">
    <table>
      <tr ng-repeat="x in names">
        <td>{{ x.Name }}</td>
        <td>{{ x.Country | uppercase }}</td>
      </tr>
    </table>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('customersCtrl', function ($scope, $http) {
      $http.get("customers.html")
        .then(function (response) { $scope.names = response.data.records; });
    });
  </script>

```



```
</body>
</html>
```

نشان دادن اندیس جدول(\$index)

به منظور نمایش دادن اندیس جدول، لازم است یک تگ **<td>** همراه با **\$index** اضافه نمایید:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
  <style>
    table, th, td {
      border: 1px solid grey;
      border-collapse: collapse;
      padding: 5px;
    }
    table tr:nth-child(odd) {
      background-color: #f1f1f1;
    }
    table tr:nth-child(even) {
      background-color: #ffffff;
    }
  </style>
</head>
<body>
  <div ng-app="myApp" ng-controller="customersCtrl">
    <table>
      <tr ng-repeat="x in names">
        <td>{{ $index + 1 }}</td>
        <td>{{ x.Name }}</td>
        <td>{{ x.Country }}</td>
      </tr>
    </table>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('customersCtrl', function ($scope, $http) {
      $http.get("http://www.w3schools.com/angular/customers.php")
        .then(function (response) { $scope.names = response.data.records; });
    });
  </script>
</body>
</html>
```

استفاده از \$odd و \$even

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
  <style>
    table, td {
      border: 1px solid grey;
      border-collapse: collapse;
      padding: 5px;
    }
  </style>
</head>
<body>
  <div ng-app="myApp" ng-controller="customersCtrl">
    <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>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('customersCtrl', function ($scope, $http) {
      $http.get("customers.html")
        .then(function (response) { $scope.names = response.data.records; });
    });
  </script>
</body>
</html>
```

Twitter Bootstrap و AngularJS

Bootstrap یک صفحه ی سبک دهی (style sheet) پرطرفدار است. مقاله ی حاضر نحوه ی استفاده از **angular** به همراه **bootstrap** را به شما آموزش می دهد.

Bootstrap

برای اضافه کردن **bootstrap** به برنامه ی **AngularJS**، کد زیر را به بخش **head** از صفحه ی **HTML** خود، ضمیمه کنید:

```
<link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
```

در زیر یک مثال **HTML** کامل وجود دارد که در آن تمام دستورات **AngularJS** و **Bootstrap** شرح داده شده است:

کد HTML

```
<div class="container">
  <h3>Users</h3>
  <table class="table table-striped">
    <thead>
      <tr>
        <th>Edit</th>
        <th>First Name</th>
        <th>Last Name</th>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="user in users">
        <td>
          <button class="btn" ng-click="editUser(user.id)">
            <span class="glyphicon glyphicon-pencil"></span> Edit
          </button>
        </td>
        <td>{{ user.fName }}</td>
        <td>{{ user.lName }}</td>
      </tr>
    </tbody>
  </table>
  <hr>
  <button class="btn btn-success" ng-click="editUser('new')">
    <span class="glyphicon glyphicon-user"></span> Create New User
  </button>
  <hr>
  <form class="form-horizontal" ng-hide="hideform">
```

```

<h3 ng-show="edit">Create New User:</h3>
<h3 ng-hide="edit">Edit User:</h3>
<div class="form-group">
  <label class="col-sm-2 control-label">First Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="fName" ng-disabled="!edit" placeholder="First Name">
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Last Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="lName" ng-disabled="!edit" placeholder="Last Name">
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Password:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw1" placeholder="Password">
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Repeat:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw2" placeholder="Repeat Password">
  </div>
</div>
<hr>
<button class="btn btn-success" ng-disabled="error || incomplete">
  <span class="glyphicon glyphicon-save"></span> Save Changes
</button>
</form>
</div>

```

شرح دستورات بکار رفته در نمونه ی فوق

شرح	دستور AngularJS
یک برنامه برای المان <body> تعریف می کند.	<body ng-app
یک کنترلر برای المان <body> تعریف می کند.	body ng- <controller
به ازای هر user در users عنصر <tr> را تکرار می کند.	<tr ng-repeat

<button ng-click	هنگامی که بر روی عنصر <button> کلیک می شود، تابع editUser() صدا زده می شود.
<h3 ng-show	نمایش کردن عنصر <h3> در صورتی که edit=true باشد.
<h3 ng-hide	پنهان کردن المان <h3> در صورتی که edit مقدار true را داشته باشد.
<input ng-model	این دستور المان <input> را به برنامه ی مورد نظر bind می کند.
button ng-disabled	چنانچه error یا incomplete دارای مقدار true باشد، المان <button> را غیر فعال می کند.

تشریح کلاس های bootstrap

المان	کلاس مربوطه	شرح
<div>	container	یک ظرف یا نگهدارنده ی محتوا تعریف می کند.
<table>	table	یک جدول تعریف می کند.
<table>	table-striped	یک جدول راه راه تعریف می کند.
<button>	btn	یک دکمه تعریف می کند.
<button>	btn-success	یک دکمه ی موفقیت تعریف می کند.
	glyphicon	یک آیکن نمادین (glyph) تعریف می کند.
	glyphicon-pencil	یک آیکن مداد تعریف می کند.
	glyphicon-user	یک آیکن کاربر تعریف می کند.

یک آیکن ذخیره تعریف می کند.	glyphicon-save	
یک فرم افقی تعریف می کند.	form-horizontal	<form>
یک گروه فرم تعریف می کند.	form-group	<div>
یک control label تعریف می کند.	control-label	<label>
یک span دو ستونه مشخص می کند.	col-sm-2	<label>
یک span ده ستونه تعریف می نماید.	col-sm-10	<div>

کد جاوا اسکریپت

myUsers.js

```
angular.module('myApp', []).controller('userCtrl', function ($scope) {
    $scope.fName = "";
    $scope.lName = "";
    $scope.passw1 = "";
    $scope.passw2 = "";
    $scope.users = [
        { id: 1, fName: 'Hege', lName: "Pege" },
        { id: 2, fName: 'Kim', lName: "Pim" },
        { id: 3, fName: 'Sal', lName: "Smith" },
        { id: 4, fName: 'Jack', lName: "Jones" },
        { id: 5, fName: 'John', lName: "Doe" },
        { id: 6, fName: 'Peter', lName: "Pan" }
    ];
    $scope.edit = true;
    $scope.error = false;
    $scope.incomplete = false;
    $scope.editUser = function (id) {
        if (id == 'new') {
            $scope.edit = true;
            $scope.incomplete = true;
            $scope.fName = "";
            $scope.lName = "";
        } else {
            $scope.edit = false;
            $scope.fName = $scope.users[id - 1].fName;
```

```

    $scope.lName = $scope.users[id - 1].lName;
  }
};
$scope.$watch('passw1', function () { $scope.test(); });
$scope.$watch('passw2', function () { $scope.test(); });
$scope.$watch('fName', function () { $scope.test(); });
$scope.$watch('lName', function () { $scope.test(); });
$scope.test = function () {
  if ($scope.passw1 !== $scope.passw2) {
    $scope.error = true;
  } else {
    $scope.error = false;
  }
  $scope.incomplete = false;
  if ($scope.edit && (!$scope.fName.length ||
    !$scope.lName.length ||
    !$scope.passw1.length || !$scope.passw2.length)) {
    $scope.incomplete = true;
  }
};
});

```

شرح کد جاوا اسکریپت

Property های (خواص) شی scope	کاربرد
\$scope.fName	متغیر model (نام کاربر).
\$scope.lName	متغیر model (نام خانوادگی کاربر).
scope.passw1	متغیر model (رمز عبور کاربر ۱).
scope.passw2	متغیر model (رمز عبور کاربر ۲).
\$scope.users	متغیر model (مجموعه یا آرایه ای از کاربران).
\$scope.edit	زمانی که کاربر بر روی create user کلیک می کند، روی true تنظیم می شود.
\$scope.error	در صورتی که passw1 برابر با passw2 نباشد، روی مقدار true تنظیم می شود.

\$scope.incomplete	در صورتی که فیلدی تهی باشد روی مقدار true تنظیم می شود(hength=0).
\$scope.editUser	متغیرهای model را مقداردهی (set) می کند.
\$scope.watch	بر متغیرهای model نظارت می کند.
\$scope.test	متغیرهای مدل را از نظر خطا یا ناکامل بودن مورد بررسی قرار می دهد.

AngularJS – واکنشی و خواندن اطلاعات از پایگاه داده SQL

کدهای آورده شده در مبحث قبلی را می توان از پایگاه داده نیز خواند.

واکنشی اطلاعات از یک سرویس دهنده ی **PHP** که پایگاه داده ی **MySQL** بر روی آن مستقر است:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
  <style>
    table, th, td {
      border: 1px solid grey;
      border-collapse: collapse;
      padding: 5px;
    }
    table tr:nth-child(odd) {
      background-color: #f1f1f1;
    }
    table tr:nth-child(even) {
      background-color: #ffffff;
    }
  </style>
</head>
<body>
  <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("customers.htm")
    .then(function (response) { $scope.names = response.data.records; });
});
</script>
</body>
</html>

```

واکشی اطلاعات از یک سرویس دهنده ی **ASP.NET** که **SQL** بر روی آن مستقر است:

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
  <style>
    table, th, td {
      border: 1px solid grey;
      border-collapse: collapse;
      padding: 5px;
    }
    table tr:nth-child(odd) {
      background-color: #f1f1f1;
    }
    table tr:nth-child(even) {
      background-color: #ffffff;
    }
  </style>
</head>
<body>
  <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("customers.htm")
        .then(function (response) { $scope.names = response.data.records; });
    });
  </script>
</body>
</html>

```

نمونه هایی از کد سرور

قسمت زیر فهرستی از کدهای سرور را نمایش می دهد که از آنها جهت واکنشی اطلاعات **SQL** استفاده می شود:

استفاده از **PHP** و **MySQL**. برگرداندن **JSON**.

استفاده از **PHP** و **MS Access**. برگرداندن **JSON**.

استفاده از **ASP.NET** و **VB** و **MS Access**. بازیابی **JSON**.

استفاده از **ASP.NET** و **Razor** و **SQL Lite**. بازگرداندن **JSON**.

درخواست های HTTP از چندین سایت/cross-site

درخواست اطلاعات از سرور متفاوت (جدا از سرور صفحه ی درخواست کننده)، درخواست **HTTP** به صورت سایت متقابل اطلاق می گردد.

درخواست های **Cross-site** (درخواست **http** از چندین سایت) در وب رایج هستند. صفحات بسیاری وجود دارند که **CSS**، تصاویر و نیز پرده های (script) خود را از سرورهای دیگر می گیرند.

در مرورگرهای نوین درخواست های **cross-site** از پرده ها (script)، به منظور رعایت مسائل امنیتی، به خود همان سایت محدود شده اند.

خط زیر به منظور فراهم آوردن امکان دسترسی **cross-site** به مثال های **PHP** اضافه شده است:

```
header("Access-Control-Allow-Origin: *");
```

کد سرور PHP و MySQL

```
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
$conn = new mysqli("myServer", "myUser", "myPassword", "Northwind");
$result = $conn->query("SELECT CompanyName, City, Country FROM Customers");
```

```

$outp = "";
while($rs = $result->fetch_array(MYSQLI_ASSOC)) {
    if ($outp != "") {$outp .= ",";}
    $outp .= '{"Name":"' . $rs["CompanyName"] . "','';
    $outp .= '"City":"' . $rs["City"] . "','';
    $outp .= '"Country":"' . $rs["Country"] . "'}';
}
$outp = '{"records":["' . $outp . '']}';
$conn->close();
echo($outp);
?>

```

۲. کد سرور PHP و MS Access

```

<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=ISO-8859-1");

$conn = new COM("ADODB.Connection");
$conn->open("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source=Northwind.mdb");

$rs = $conn->execute("SELECT CompanyName, City, Country FROM Customers");

$outp = "";
while (!$rs->EOF) {
    if ($outp != "") {$outp .= ",";}
    $outp .= '{"Name":"' . $rs["CompanyName"] . "','';
    $outp .= '"City":"' . $rs["City"] . "','';
    $outp .= '"Country":"' . $rs["Country"] . "'}';
    $rs->MoveNext();
}
$outp = '{"records":["' . $outp . '']}';

$conn->close();

echo ($outp);
?>

```

۳. کد سرور ASP.NET، VB Razor و SQL Lite

```

@{
    Response.AppendHeader("Access-Control-Allow-Origin", "**")
    Response.AppendHeader("Content-type", "application/json")
    var db = Database.Open("Northwind");
    var query = db.Query("SELECT CompanyName, City, Country FROM Customers");
    var outp = ""
    var c = chr(34)
}

```

```
@foreach(var row in query)
{
if outp <> "" then outp = outp + ","
outp = outp + "{" + c + "Name" + c + ":" + c + @row.CompanyName + c + ","
outp = outp + c + "City" + c + ":" + c + @row.City + c + ","
outp = outp + c + "Country" + c + ":" + c + @row.Country + c + "}"
}
outp = "{" + c + "records" + c + ":" + outp + "}"
@outp
```

ع. کد سرور ASP.NET، VB Razor و SQL Lite

```
@{
Response.AppendHeader("Access-Control-Allow-Origin", "**")
Response.AppendHeader("Content-type", "application/json")
var db = Database.Open("Northwind");
var query = db.Query("SELECT CompanyName, City, Country FROM Customers");
var outp = ""
var c = chr(34)
}
@foreach(var row in query)
{
if outp <> "" then outp = outp + ","
outp = outp + "{" + c + "Name" + c + ":" + c + @row.CompanyName + c + ","
outp = outp + c + "City" + c + ":" + c + @row.City + c + ","
outp = outp + c + "Country" + c + ":" + c + @row.Country + c + "}"
}
outp = "{" + c + "records" + c + ":" + outp + "}"
@outp
```

مدل شی گرای سند – HTML DOM در AngularJS

AngularJS دارای دستورهایی برای پیوند دادن (**bind** کردن) داده های برنامه های **Angular** به خصیصه ها یا **attribute** های عناصر **HTML DOM** است.

دستور ng-disabled

دستور **ng-disabled** داده های برنامه **Angular** را به خصیصه ی **disabled** عناصر **HTML** ، پیوند می دهد.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="" ng-init="mySwitch=true">
    <p>
      <button ng-disabled="mySwitch">Click Me!</button>
    </p>
    <p>
      <input type="checkbox" ng-model="mySwitch" />Button
    </p>
    <p>
      {{ mySwitch }}
    </p>
  </div>
</body>
</html>
```

شرح برنامه ی بالا:

دستور **ng-disabled** داده ی **mySwitch** برنامه را به خصیصه ی **disabled** دکمه ی **HTML** متصل (**bind**) می کند.

دستور **ng-model** مقدار المان **checkbox** را به مقدار **mySwitch** متصل می کند.

در صورتی که مقدار **mySwitch** برابر با **true** شود، دکمه ی مورد نظر غیرفعال می شود:

```
<p>
<button disabled>Click Me!</button>
</p>
```

چنانچه مقدار **mySwitch** برابر با **false** باشد، دکمه ی مورد نظر، غیرفعال نمی شود:

```
<p>
<button>Click Me!</button>
</p>
```

دستور ng-show

دستور **ng-show** یک عنصر **HTML** را نمایان می کند یا پنهان می سازد.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="">
    <p ng-show="true">I am visible.</p>
    <p ng-show="false">I am not visible.</p>
  </div>
</body>
</html>
```

دستور **ng-show** با توجه به مقدار **ng-show** یک عنصر **HTML** را محو کرده یا نمایش می دهد.

شما می توانید از هر عبارتی که **true** یا **false** را برمی گرداند، استفاده نمایید.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="" ng-init="hour=13">
    <p ng-show="hour > 12">I am visible.</p>
  </div>
</body>
</html>
```

دستور ng-hide

دستور **ng-hide** یک المان **HTML** را پنهان کرده و یا نمایان می سازد.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
```

```
<body>
  <div ng-app="">
    <p ng-hide="true">I am not visible.</p>
    <p ng-hide="false">I am visible.</p>
  </div>
</body>
</html>
```

رخدادها در angularJS

دستور ng-click

دستور **ng-click** یک رخداد تعریف می کند که به محض کلیک بر روی دکمه توسط کاربر، اجرا می شود.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="myCtrl">
    <button ng-click="count = count + 1">Click Me!</button>
    <p>{{ count }}</p>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function ($scope) {
      $scope.count = 0;
    });
  </script>
</body>
</html>
```

پنهان سازی المان های HTML

از دستور **ng-hide** برای تنظیم قابلیت رویت و نمایش (**visibility**) یک قسمت از برنامه استفاده می شود.

مقدار **ng-hide="true"** باعث می شود که یک عنصر **HTML** محو شود.

مقدار **ng-hide="false"** باعث می شود یک المان **HTML** قابل رویت گردد.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
```

```

<script src="Angular.js">/script>
</head>
<body>
  <div ng-app="myApp" ng-controller="personCtrl">
    <button ng-click="toggle()">Hide user</button>
    <p ng-hide="myVar">
      First Name: <input type="text" ng-model="firstName"><br>
      Last Name: <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>
</body>
</html>

```

شرح برنامه:

اولین قسمت **personController** دقیقا مانند آنچه در مبحث کنترلرها یاد گرفتید است.

برنامه ی مورد نظر دارای یک خاصیت (**property**) پیش فرض (یک متغیر) است: **\$scope.myVar=false**

دستور **ng-hide** نمایش یافتن یا محو شدن عنصر **<p>** به همراه دو فیلد **input** را بر اساس مقدار متغیر

myVar که می تواند **true** یا **false** باشد، تعیین می کند.

تابع **toggle()** باعث می شود متغیر **myVar** بین **true** و **false** تغییر وضعیت دهد.

دستور **ng-hide="true"** باعث می شود که عنصر مورد نظر، پنهان شود.

نمایش دادن عناصر HTML

همچنین از دستور **ng-show** می توان برای تنظیم نمایش و قابلیت رویت (**visibility**) یک قسمت از برنامه

بهره گرفت.

دستور **ng-show="false"** المان HTML را محو می کند.

دستور `ng-show="true"` یک المان **HTML** را نمایان می کند.

در زیر نمونه ای مانند مثال بالا را می بینید با این فرق که بجای `ng-hide` از `ng-show` استفاده شده:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="personCtrl">
    <button ng-click="toggle()">Hide user</button>
    <p ng-show="myVar">
      First Name: <input type="text" ng-model="person.firstName"><br>
      Last Name: <input type="text" ng-model="person.lastName"><br><br>
      Full Name: {{person.firstName + " " + person.lastName}}
    </p>
  </div>
<script>
  var app = angular.module('myApp', []);
  app.controller('personCtrl', function ($scope) {
    $scope.person = {
      firstName: "John",
      lastName: "Doe"
    };
    $scope.myVar = true;
    $scope.toggle = function () {
      $scope.myVar = !$scope.myVar;
    };
  });
</script>
</body>
</html>
```

فرم ها در AngularJS

فرم در **AngularJS** عبارت است از یک مجموعه از کنترل ها که ورودی دریافت می کنند.

کنترل های HTML

المان هایی که در **HTML** ورودی می پذیرند (**input elements**)، عبارتند از:

[المان input](#)[المان select](#)[المان های button](#)[المان های textarea](#)

فرم های HTML

فرم های **HTML** کنترل های **HTML** را در یک مجموعه و پکیج واحد گنجانده و گروه بندی می کند.

نمونه ای از فرم AngularJS

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <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>
</body>
</html>
```

خصوصیه ی **novalidate** در **HTML5** یک امکان نوین می باشد. خصوصیه ی مزبور هرگونه اعتبارسنجی پیش فرض مرورگر را غیرفعال می سازد.

شرح برنامه:

دستور **ng-app** همان طور که می دانید برنامه ی **AngularJS** را تعریف می کند.

دستور **ng-controller** هم که کنترلگر برنامه را تعریف می کند.

دستور **ng-model** دو المان های **input** را به شی **user** در مدل **bind** می کند.

متد **formCtrl** مقادیر اولیه را در شی **master** تنظیم می کند و نیز متد **reset()** را اعلان می کند.

reset() شی **user** را با **master** برابر قرار می دهد.

دستور **ng-click**، تنها در صورتی که دکمه ی مورد نظر کلیک شود متد **reset** را صدا می زند.

خصوصیه **(attribute) novalidate** برای برنامه ی حاضر مورد نیاز نمی باشد، اما اغلب از آن در فرم ها به منظور بازنویسی **(override)** اعتبارسنجی معمول **(validation) HTML** استفاده می شود.

اعتبارسنجی ورودی در AngularJS

کنترل ها و فرم های دریافت کننده ی ورودی این قابلیت را دارند که داده های ورودی را اعتبارسنجی کنند.

Input Validation

در مبحث پیشین مطالبی را در مورد فرم ها و کنترل ها آموختید.

فرم ها و کنترل ها در **AngularJS**، می توانند سرویس های اعتبارسنجی ارائه داده و کاربران را از وجود ورودی های غیرمجاز **(invalid)** آگاه سازند.

توجه: اعتبارسنجی در سمت سرویس گیرنده به تنهایی قادر به تامین امنیت ورودی های کاربر نیست. اعتبارسنجی در سمت سرویس دهنده نیز مورد نیاز می باشد.

کد برنامه ی نمونه

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <h2>Validation Example</h2>
  <form ng-app="myApp" ng-controller="validateCtrl"
    name="myForm" novalidate>
    <p>
      Username:<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.$error.required">Username is required.</span>
      </span>
    </p>
    <p>
      Email:<br>
      <input type="email" name="email" ng-model="email" required>
      <span style="color:red" ng-show="myForm.email.$dirty && myForm.email.$invalid">
        <span ng-show="myForm.email.$error.required">Email is required.</span>
        <span ng-show="myForm.email.$error.email">Invalid email address.</span>
      </span>
    </p>
    <p>
      <input type="submit"
        ng-disabled="myForm.user.$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>

```

نکته: خصیصه ی **novalidate** جهت غیرفعال کردن اعتبارسنجی پیش فرض مرورگر بکار می رود.

شرح مثال بالا:

دستور **ng-model** عناصر ورودی را به مدل **bind** می کند.

شی **model** دارای دو خاصیت می باشد: **user** و **email** .

به خاطر دستور **ng-show**، عناصر **span** با این خاصیت **color:red** فقط زمانی نمایش داده می شوند که **user** یا **email** مساوی با **\$dirty** یا **\$invalid** باشند.

شرح	خاصیت (property)
کاربر با فیلد مورد نظر تعامل داشته است.	\$dirty
محتوای فیلد مورد نظر مجاز می باشد.	\$valid
ورودی فیلد مورد نظر غیرمجاز می باشد.	\$invalid
کاربر هنوز با فیلد تعامل نداشته.	\$pristine

رابط برنامه سازی کاربردی (API) در AngularJS

API سرواژه ی **Interface Programming Application** (رابط برنامه سازی کاربردی) می باشد.

API ی سراسری AngularJS

API سراسری **AngularJS**، یک سری توابع جاوااسکریپت هستند که از آن ها جهت انجام کارهایی همچون موارد زیر استفاده می شود:

۱. مقایسه کردن اشیا
۲. تکرار کردن اشیا
۳. تبدیل کردن اطلاعات

توابع **API** سراسری **AngularJS**، با از استفاده از شی **angular** قابل دسترسی می باشند.

در زیر فهرستی از توابع **API** متعارف **AngularJS** برای شما به نمایش گذاشته شده است:

API ی مربوطه	شرح
angular.lowercase()	تبدیل یک رشته به حروف کوچک توسط این API صورت می پذیرد.
angular.uppercase()	از این API جهت تبدیل یک رشته به حروف بزرگ استفاده می شود.
angular.isString()	این API، در صورتی که reference مورد نظر یک رشته باشد، مقدار true بازگردانی می نماید.
angular.isNumber()	این API، در صورتی که reference مورد نظر عدد باشد، true بازایی می کند.

نمونه ای از کاربرد تابع angular.lowercase()

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="myCtrl">
    <p>{{ x1 }}</p>
    <p>{{ x2 }}</p>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function ($scope) {
      $scope.x1 = "JOHN";
      $scope.x2 = angular.lowercase($scope.x1);
    });
  </script>
</body>
</html>

```

نمونه ای از تابع angular.uppercase()

```

<!DOCTYPE html>
<html>
<head>
  <title>
</title>
  <script src="Angular.js"></script>

```

```

</head>
<body>
  <div ng-app="myApp" ng-controller="myCtrl">
    <p>{{ x1 }}</p>
    <p>{{ x2 }}</p>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function ($scope) {
      $scope.x1 = "John";
      $scope.x2 = angular.uppercase($scope.x1);
    });
  </script>
</body>
</html>

```

نمونه ای از تابع angular.isString()

```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="myCtrl">
    <p>{{ x1 }}</p>
    <p>{{ x2 }}</p>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function ($scope) {
      $scope.x1 = "JOHN";
      $scope.x2 = angular.isString($scope.x1);
    });
  </script>
</body>
</html>

```

تابع angular.isNumber()

```

<!DOCTYPE html>
<html>
<head>
  <title></title>

```

```

<script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="myCtrl">
    <p>{{ x1 }}</p>
    <p>{{ x2 }}</p>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function ($scope) {
      $scope.x1 = "JOHN";
      $scope.x2 = angular.isNumber($scope.x1);
    });
  </script>
</body>
</html>

```

W3.CSS و AngularJS

شما می توانید به آسانی از **w3.css** و **AngularJS** با هم استفاده کنید. در این مقاله قصد داریم نحوه انجام این کار را به شما آموزش دهیم.

W3.CSS

برای آنکه فایل **W3.CSS** شما با برنامه کاربردی **AngularJS** شما همراه شود باید خط زیر را به قسمت **head** خود اضافه کنید.

```
<link rel="stylesheet" href="http://www.w3schools.com/lib/w3.css">
```

در زیر یک مثال **HTML** کامل را با استفاده از **AngularJS** و کلاس **w3.css** را توضیح می دهیم.

کد HTML

```

<body ng-app="myApp" ng-controller="userCtrl">
  <div class="w3-container">
    <h3>Users</h3>
    <table class="w3-table w3-bordered w3-striped">
      <tr>
        <th>Edit</th>
        <th>First Name</th>
        <th>Last Name</th>
      </tr>
      <tr ng-repeat="user in users">
        <td>

```



```

        <button type="button" class="w3-btn w3-ripple" ng-click="editUser(user.id)">#9998; Edit</button>
    </td>
    <td>{{ user.fName }}</td>
    <td>{{ user.lName }}</td>
</tr>
</table>
<br>
<button type="button" class="w3-btn w3-green w3-ripple" ng-click="editUser('new')">#9998; Create New
User</button>
<form ng-hide="hideform">
    <h3 ng-show="edit">Create New User:</h3>
    <h3 ng-hide="edit">Edit User:</h3>
    <label>First Name:</label>
    <input class="w3-input w3-border" type="text" ng-model="fName" ng-disabled="!edit" placeholder="First Name">
    <br>
    <label>Last Name:</label>
    <input class="w3-input w3-border" type="text" ng-model="lName" ng-disabled="!edit" placeholder="Last Name">
    <br>
    <label>Password:</label>
    <input class="w3-input w3-border" type="password" ng-model="passw1" placeholder="Password">
    <br>
    <label>Repeat:</label>
    <input class="w3-input w3-border" type="password" ng-model="passw2" placeholder="Repeat Password">
    <br>
    <button type="button" class="w3-btn w3-green w3-ripple" ng-disabled="error || incomplete">#10004; Save
Changes</button>
</form>
</div>
<script src="http://www.tahlildadeh.com/Jquery/Angular/myUsers.js"></script>
</body>

```

توضیح Directive های استفاده شده در مثال بالا

توضیحات	AngularJS Directive
یک برنامه کاربردی برای المان <body> تعریف می کند.	<body ng-app>
یک کنترلر برای المان <body> تعریف می کند.	<body ng-controller>
به ازای هر کاربر، یک <tr> در کاربرها تکرار می کند.	<tr ng-repeat>
فراخوانی تابع editUser() وقتی <button> کلیک می شود.	<button ng-click>
هرگاه edit=true باشد، <h3> ها را نمایش می دهد.	<h3 ng-show>
هرگاه hideform=true و وقتی edit=true باشد <h3> را مخفی می کند.	<h3 ng-hide>

المان <code><input></code> را به برنامه کاربردی متصل می کند.	<code><input ng-model></code>
اگر خطایی رخ دهد و یا <code>incomplete=true</code> باشد المان <code><button></code> را غیر فعال می کند	<code><button ng-disabled></code>

توصیف کلاس های W3.CSS

توضیحات	Class	Element
محتوای Container	w3-container	<code><div></code>
جدول	w3-table	<code><table></code>
حاشیه جدول	w3-bordered	<code><table></code>
جدول راه راه	w3-striped	<code><table></code>
کلید	w3-btn	<code><button></code>
کلید سبز رنگ	w3-green	<code><button></code>
افکت موجی در هنگام کلیک کردن	w3-ripple	<code><button></code>
مقدار ورودی	w3-input	<code><input></code>
لبه در مقدار ورودی	w3-border	<code><input></code>

کد JavaScript

```
angular.module('myApp', []).controller('userCtrl', function ($scope) {
    $scope.fName = "";
    $scope.lName = "";
    $scope.passw1 = "";
    $scope.passw2 = "";
    $scope.users = [
        { id: 1, fName: 'Hege', lName: "Pege" },
        { id: 2, fName: 'Kim', lName: "Pim" },
        { id: 3, fName: 'Sal', lName: "Smith" },
        { id: 4, fName: 'Jack', lName: "Jones" },
        { id: 5, fName: 'John', lName: "Doe" },
        { id: 6, fName: 'Peter', lName: "Pan" }
    ];
    $scope.edit = true;
    $scope.error = false;
});
```

```

$scope.incomplete = false;
$scope.hideform = true;
$scope.editUser = function (id) {
    $scope.hideform = false;
    if (id == 'new') {
        $scope.edit = true;
        $scope.incomplete = true;
        $scope.fName = "";
        $scope.lName = "";
    } else {
        $scope.edit = false;
        $scope.fName = $scope.users[id - 1].fName;
        $scope.lName = $scope.users[id - 1].lName;
    }
};
$scope.$watch('passw1', function () { $scope.test(); });
$scope.$watch('passw2', function () { $scope.test(); });
$scope.$watch('fName', function () { $scope.test(); });
$scope.$watch('lName', function () { $scope.test(); });
$scope.test = function () {
    if ($scope.passw1 !== $scope.passw2) {
        $scope.error = true;
    } else {
        $scope.error = false;
    }
    $scope.incomplete = false;
    if ($scope.edit && (!$scope.fName.length ||
    !$scope.lName.length ||
    !$scope.passw1.length || !$scope.passw2.length)) {
        $scope.incomplete = true;
    }
};
});

```

توضیح کدهای Java Script

علت استفاده	خاصیت Scope
نام کاربر	<code>\$scope.fName</code>
نام خانوادگی کاربر	<code>\$scope.lName</code>
پسورد ۱	<code>\$scope.passw1</code>
پسورد ۲	<code>\$scope.passw2</code>
آرایه ای از کاربران	<code>\$scope.users</code>

\$scope.edit	هرگاه کاربر روی کلید Creat User کلیک کند مقدار آن true می شود.
\$scope.hideform	هرگاه کاربر روی کلید Edit کلیک کند مقدار آن true می شود.
\$scope.error	هرگاه passw1 و passw2 یکسان نباشند مقدار آن true می شود.
\$scope.incomplete	هرگاه مقداری خالی باشد (length=0) مقدار آن true می شود.
\$scope.editUser	مجموعه از متغیرهای مدل
\$scope.\$watch	متغیرهای مدل Watch
\$scope.test	متغیرهای تست مدل برای خطاها و مقدارهای کامل نشده

Includes در AngularJS

با استفاده از **AngularJS** می توان فایل های **HTML** را در **HTML** اضافه کرد.

در **AngularJS** می توان به واسطه ی دستور **ng-include** محتویات **HTML** را در **HTML** تزریق کنید:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body ng-app="">
  <div ng-include="myFile.htm"></div>
</body>
</html>
```

تزریق کد AngularJS به همراه HTML

فایل های **HTML** ای که با استفاده از دستور **ng-include** اضافه می کنید، همچنین می توانند حاوی کد

AngularJS باشند:

فایل myTable.htm:

```
<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>
```

فایل myTable.htm را به صفحه ی وب خود الحاق کنید. خواهی دید که تمامی کدهای **AngularJS** (به

ضمیمه ی تمامی کدهای موجود در فایل اضافه شده به صفحه ی وب) اجرا خواهد شد:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="Angular.js"></script>
</head>
<body>
  <div ng-app="myApp" ng-controller="customersCtrl">
    <div ng-include="myTable.htm"></div>
  </div>
  <script>
    var app = angular.module('myApp', []);
    app.controller('customersCtrl', function ($scope, $http) {
      $http.get("customers.html")
        .then(function (response) { $scope.names = response.data.records; });
    });
  </script>
  <p>The HTML, and AngularJS code, for this table are located in the file "myTable.htm".</p>
</body>
</html>
```

کد های **HTML** و **AngularJS** که یان جدول را تعریف کرده و نمایش می دهند در فایل الحاقی "myTable.htm" قرار دارد.

انیمیشن AngularJS

انیمیشن چیست؟

هرگاه تغییر وضعیت در یک المان **HTML** صورت گیرد، این حالت را به اصطلاح انیمیشن می گویند.

مثال: تغییر وضعیت **DIV** با کلیک بر روی چک باکس به صورت زیر است.

سورس این برنامه به صورت رایگان در انتهای این مقاله در دسترس می باشد.

به چه چیزهایی نیاز داریم؟

برای آماده کردن برنامه کاربردی خود به حالت انیمیشن، شما باید کتابخانه **AngularJS Animate** را به قسمت **head** برنامه خود اضافه کنید.

```
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-animate.js"></script>
```

سپس شما باید به ماژول **ngAnimate** متصل کنید.

```
<body ng-app="ngAnimate">
```

و یا اگر برنامه شما نام خاص دارد، می توانید **ngAnimate** را به طور وابسته به برنامه خود اضافه کنید:

سورس این برنامه به صورت رایگان در انتهای این مقاله در دسترس می باشد.

ngAnimate چه کاری انجام می دهد؟

این ماژول، کلاس هایی را اضافه و یا حذف می کند. ماژول **ngAnimate** المان های **HTML** شما را حرکت نمی دهد اما هرگاه **ngAnimate** رخداد خاصی را اعلان کند، به عنوان مثال محو یا ظاهر کردن یک المان **HTML**، المان چند کلاس پیش فرض را که می تواند انیمیشن تولید کند، را دریافت می کند. در زیر **Directive** هایی که مسئول اضافه و حذف کلاس ها در **AngularJS** هستند معرفی می گردند:

- ng-show
- ng-hide
- ng-class
- ng-view
- ng-include
- ng-repeat
- ng-if
- ng-switch
- ng-show
- ng-hide
- ng-class
- ng-view
- ng-include
- ng-repeat
- ng-if

ng-show و **ng-hide** مقدار کلاس **ng-hide** را اضافه یا حذف می کند.

دایرکتیو های دیگر در هنگام بالا آمدن **DOM** کلاس **ng-enter** و یک اتریبوت **ng-leave** هنگام حذف از **DOM**، را مقدار می دهد.

همچنین دایرکتیو **ng-repeat** مقدار کلاس **ng-move** را زمانی که المان **HTML** تغییر مکان می دهد اضافه می کند. به علاوه، در هنگام اجرای انیمیشن المان **HTML**، یک سری از کلاس ها را مقداردهی می کند و هرگاه اجرای انیمیشن به پایان رسید، آنها را حذف می کند.

- **ng-animate**
- **ng-hide-animate**
- **ng-hide-add** (المان مخفی خواهد شد)
- **ng-hide-remove** (المان نشان داده خواهد شد)
- **ng-hide-add-active** (المان مخفی خواهد شد)
- **ng-hide-remove-active** (المان نشان داده خواهد شد)

انیمیشن به استفاده از CSS

ما می توانیم از خاصیت انتقال (**Css(Transition**) و خاصیت انیمیشن (**Css(Animation**) ها برای متحرک سازی المان های **HTML** استفاده کنیم. در این مقاله این روش به شما آموزش داده می شود.

خاصیت انتقال (CSS (Transition) ها

این خاصیت **CSS** ها تغییر ویژگی **CSS** را به راحتی با تغییر مقدار و مقداردهی زمانی به شما می دهد.

مثال : وقتی یک **DIV**، کلاس **ng-hide** می گیرد، در ۵/۰ ثانیه ارتفاع از ۱۰۰px به ۰ می رسد:

سورس این برنامه به صورت رایگان در انتهای این مقاله در دسترس می باشد.

خاصیت CSS Animation ها

خاصیت **CSS** ها تغییر ویژگی **CSS** را به راحتی با تغییر مقدار و مقداردهی زمانی به شما می دهد.

مثال : وقتی یک **DIV** کلاس **ng-hide** می گیرد، انیمیشن **myChange** اجرا می شود و به آرامی ارتفاع از 100px تا ۰ تغییر می کند.

برنامه ی تحت وب AngularJS

با مطالبی آموزش هایی که تاکنون آموخته ایم، حال می توانیم یک برنامه ی تک صفحه ای (SPA) را به طور واقعی بسازیم.

نمونه ای از یک برنامه ی AngularJS

اکنون به اندازه ی کافی مطلب جهت ایجاد کردن اولین برنامه ی AngularJS فراگرفته اید.

کد این برنامه را در زیر مشاهده می کنید:

```
<body ng-app="myNoteApp" ng-controller="myNoteCtrl">
  <h2>My Note</h2>
  <textarea ng-model="message" cols="40" rows="10"></textarea>
  <p>
    <button ng-click="save()">Save</button>
    <button ng-click="clear()">Clear</button>
  </p>
  <p>Number of characters left: <span ng-bind="left()"></span></p>
  <script src="myNoteApp.js"></script>
  <script src="myNoteCtrl.js"></script>
</body>
```

فایل حاوی برنامه "myNoteApp.js":

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

فایل دربردارنده ی کنترلر "myNoteCtrl.js":

```
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"); };
});
```

کتابخانه های برنامه ی شما به صفحه ی مورد نظر اضافه می شوند:

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

عنصر html که دربردارنده ی برنامه ی AngularJS است : myNoteApp"=ng-app"


```
<html ng-app="myNoteApp">
```

تگ `<div>` در سند HTML حوزه ی (scope) یک کنترلر می باشد: `myNoteCtrl"=ng-controller"`

```
<div ng-controller="myNoteCtrl">
```

دستور `ng-model` محتوای یک `<textarea>` را به متغیر کنترلر `message`، `bind` می کند:

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

دو رخداد `ng-click` توابع کنترلر `clear()` و `save()` را صدا می زنند:

```
<button ng-click="save()">Save</button>
```

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

دستور `ng-bind` تابع کنترلر `left()` را به یک `` که حاوی کاراکترهای باقی مانده است، `bind` یا متصل می کند:

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

کتابخانه های برنامه ی شما به صفحه ی مورد نظر اضافه شده اند (پس از کتابخانه):

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

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

اسکلت و ساختار برنامه ی کاربردی AngularJS

در بالا، یک ساختار برنامه ی کاربردی واقعی تک صفحه ای را نظاره گر هستید.

المان `<html>` ظرف نگهدارنده ی برنامه ی کاربردی AngularJS می باشد. (`ng-app=`)

المان `<div>` حوزه ی (scope) کنترلر AngularJS را مشخص می کند. (`ng-controller=`)

می توان چندین کنترلر در یک برنامه ی AngularJS داشت.

App file (`my...App.js`) کد ماژول برنامه را تعریف می کند.

یک یا چند فایل کنترلر (my...Ctrl.js) کد کنترلر را تعریف می کنند.

چکیده - برنامه چگونه کار می کند؟

دستور **ng-app** در **root element** (عنصر آغازین یا به عبارتی ریشه) برنامه قرار گرفته است.

برای یک برنامه ی تک صفحه ای، معمولا ریشه یا عنصر آغازین برنامه ی همان عنصر **<html>** است.

یک یا چند دستور **ng-controller** در واقع کنترلرهای برنامه ی مورد نظر را تعریف می کنند. هر کنترلر حوزه ی (**scope**) مختص خود را دارد که عبارت است از عنصر **HTML** ای که در آنجا تعریف شده است.

AngularJS به محض اجرای رخداد **HTML DOMContentLoaded** به صورت خودکار راه اندازی می شود. در صورت وجود دستور **ng-app**، **AngularJS** ماژول های نام گذاری شده در دستور نام برده را بارگذاری می کند و سپس برنامه **DOM** را با **ng-app** به عنوان ریشه ی برنامه کامپایل می کند .

ریشه ی برنامه (**application root**) می تواند کل صفحه و یا یک قسمت کوچکتر از آن باشد. هرچه که این قسمت کوچکتر باشد، ترجمه و اجرای برنامه سریع تر صورت می پذیرد.

در پایان ضمن تشکر از انتخاب شما، امیدواریم مطالب این کتاب برای شما مفید بوده باشد.

علاوه بر این می توانید پیشنهادات و انتقادات خود را از طریق رایانامه Book.tahlildadeh@gmail.com با ما در میان بگذارید.