

AngularJS Tutorial

W3SCHOOLS.com



AngularJS extends HTML with new attributes.

AngularJS is perfect for Single Page Applications (SPAs).

AngularJS is easy to learn.

This Tutorial

This tutorial is specially designed to help you learn AngularJS as quickly and efficiently as possible.

First, you will learn the basics of AngularJS: directives, expressions, filters, modules, and controllers.

Then you will learn everything else you need to know about AngularJS:

Events, DOM, Forms, Input, Validation, Http, and more.

Try it Yourself Examples in Every Chapter

In every chapter, you can edit the examples online, and click on a button to view the result.

AngularJS Example

```
<!DOCTYPE html>
<html lang="en-US">
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<body>

<div ng-app="">
  <p>Name : <input type="text" ng-model="name"></p>
  <h1>Hello {{name}}</h1>
</div>

</body>
</html>
```

What You Should Already Know

Before you study AngularJS, you should have a basic understanding of:

- HTML
- CSS
- JavaScript

AngularJS History

AngularJS version 1.0 was released in 2012.

Miško Hevery, a Google employee, started to work with AngularJS in 2009.

The idea turned out very well, and the project is now officially supported by Google.

AngularJS is a **JavaScript framework**. It can be added to an HTML page with a `<script>` tag.

AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.

AngularJS is a JavaScript Framework

AngularJS is a JavaScript framework. It is a library written in JavaScript.

AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

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

AngularJS Extends HTML

AngularJS extends HTML with **ng-directives**.

The **ng-app** directive defines an AngularJS application.

The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.

The **ng-bind** directive binds application data to the HTML view.

AngularJS Example

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

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>

</body>
</html>
```

Example explained:

AngularJS starts automatically when the web page has loaded.

The **ng-app** directive tells AngularJS that the <div> element is the "owner" of an AngularJS **application**.

The **ng-model** directive binds the value of the input field to the application variable **name**.

The **ng-bind** directive binds the **innerHTML** of the <p> element to the application variable **name**.

AngularJS Directives

As you have already seen, AngularJS directives are HTML attributes with an **ng** prefix.

The **ng-init** directive initialize AngularJS application variables.

AngularJS Example

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

Alternatively with valid HTML:

AngularJS Example

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



You can use **data-ng-**, instead of **ng-**, if you want to make your page HTML valid.

You will learn a lot more about directives later in this tutorial.

AngularJS Expressions

AngularJS expressions are written inside double braces: **{{ expression }}**.

AngularJS will "output" data exactly where the expression is written:

AngularJS Example

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

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

```
</div>

</body>
</html>
```

AngularJS expressions bind AngularJS data to HTML the same way as the **ng-bind** directive.

AngularJS Example

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

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p>{{name}}</p>
</div>

</body>
</html>
```

You will learn more about expressions later in this tutorial.

AngularJS Applications

AngularJS **modules** define AngularJS applications.

AngularJS **controllers** control AngularJS applications.

The **ng-app** directive defines the application, the **ng-controller** directive defines the controller.

AngularJS Example

```
<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>
```

AngularJS modules define applications:

AngularJS Module

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

AngularJS controllers control applications:

AngularJS Controller

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

You will learn more about modules and controllers later in this tutorial.

AngularJS Expressions

AngularJS binds data to HTML using **Expressions**.

AngularJS Expressions

AngularJS expressions are written inside double braces: **{{ expression }}**.

AngularJS expressions binds data to HTML the same way as the **ng-bind** directive.

AngularJS will "output" data exactly where the expression is written.

AngularJS expressions are much like **JavaScript expressions**: They can contain literals, operators, and variables.

Example **{{ 5 + 5 }}** or **{{ firstName + " " + lastName }}**

AngularJS Example

```
<!DOCTYPE html>  
<html>  
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>  
<body>  
  
<div ng-app="">  
  <p>My first expression: {{ 5 + 5 }}</p>  
</div>  
  
</body>  
</html>
```

If you remove the **ng-app** directive, HTML will display the expression as it is, without solving it:

AngularJS Example

```
<!DOCTYPE html>  
<html>  
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>  
<body>  
  
<div>  
  <p>My first expression: {{ 5 + 5 }}</p>  
</div>  
  
</body>  
</html>
```

AngularJS Numbers

AngularJS numbers are like JavaScript numbers:

AngularJS Example

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

Same example using ng-bind:

AngularJS Example

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



Using **ng-init** is not very common. You will learn a better way to initialize data in the chapter about controllers.

AngularJS Strings

AngularJS strings are like JavaScript strings:

AngularJS Example

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

Same example using ng-bind:

AngularJS Example

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

AngularJS Objects

AngularJS objects are like JavaScript objects:

AngularJS Example

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

```
</div>
```

Same example using ng-bind:

AngularJS Example

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

AngularJS Arrays

AngularJS arrays are like JavaScript arrays:

AngularJS Example

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

Same example using ng-bind:

AngularJS Example

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

AngularJS Expressions vs. JavaScript Expressions

Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.

Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.

AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.

AngularJS Directives

AngularJS lets you extend HTML with new attributes called **Directives**.

AngularJS Directives

AngularJS directives are extended HTML attributes with the prefix **ng-**.

The **ng-app** directive initializes an AngularJS application.

The **ng-init** directive initializes application data.

The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.

AngularJS Example

```
<div ng-app="" ng-init="firstName='John'">
<p>Name: <input type="text" ng-model="firstName"></p>
<p>You wrote: {{ firstName }}</p>
</div>
```

The **ng-app** directive also tells AngularJS that the `<div>` element is the "owner" of the AngularJS application.

Data Binding

The `{{ firstName }}` expression, in the example above, is an AngularJS data binding expression.

Data binding in AngularJS synchronizes AngularJS expressions with AngularJS data.

`{{ firstName }}` is synchronized with `ng-model="firstName"`.

In the next example two text fields are synchronized with two `ng-model` directives:

AngularJS Example

```
<div ng-app="" ng-init="quantity=1;price=5">
Quantity: <input type="number" ng-model="quantity">
Costs: <input type="number" ng-model="price">
Total in dollar: {{ quantity * price }}
</div>
```



Using **ng-init** is not very common. You will learn how to initialize data in the chapter about controllers.

Repeating HTML Elements

The **ng-repeat** directive repeats an HTML element:

AngularJS Example

```
<div ng-app="" ng-init="names=['Jani','Hege','Kai']">
<ul>
<li ng-repeat="x in names">
  {{ x }}
</li>
</ul>
</div>
```

The **ng-repeat** directive used on an array of objects:

AngularJS Example


```
<div ng-app="" ng-init="names=[
{name:'Jani',country:'Norway'},
{name:'Hege',country:'Sweden'},
{name:'Kai',country:'Denmark'}]">

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

</div>
```



AngularJS is perfect for database CRUD (Create Read Update Delete) applications. Just imagine if these objects were records from a database.

The ng-app Directive

The **ng-app** directive defines the **root element** of an AngularJS application.

The **ng-app** directive will **auto-bootstrap** (automatically initialize) the application when a web page is loaded.

Later you will learn how **ng-app** can have a value (like `ng-app="myModule"`), to connect code modules.

The ng-init Directive

The **ng-init** directive defines **initial values** for an AngularJS application.

Normally, you will not use `ng-init`. You will use a controller or module instead.

You will learn more about controllers and modules later.

The ng-model Directive

The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.

The **ng-model** directive can also:

- Provide type validation for application data (number, email, required).
- Provide status for application data (invalid, dirty, touched, error).
- Provide CSS classes for HTML elements.
- Bind HTML elements to HTML forms.

The ng-repeat Directive

The **ng-repeat** directive **clones HTML elements** once for each item in a collection (in an array).

AngularJS Controllers

AngularJS controllers **control the data** of AngularJS applications.

AngularJS controllers are regular **JavaScript Objects**.

AngularJS Controllers

AngularJS applications are controlled by controllers.

The **ng-controller** directive defines the application controller.

A controller is a **JavaScript Object**, created by a standard JavaScript **object constructor**.

AngularJS Example

```
<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>
```

Application explained:

The AngularJS application is defined by **ng-app="myApp"**. The application runs inside the `<div>`.

The **ng-controller="myCtrl"** attribute is an AngularJS directive. It defines a controller.

The **myCtrl** function is a JavaScript function.

AngularJS will invoke the controller with a **\$scope** object.

In AngularJS, `$scope` is the application object (the owner of application variables and functions).

The controller creates two properties (variables) in the scope (**firstName** and **lastName**).

The **ng-model** directives bind the input fields to the controller properties (`firstName` and `lastName`).

Controller Methods

The example above demonstrated a controller object with two properties: `lastName` and `firstName`.

A controller can also have methods (variables as functions):

AngularJS Example

```
<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>
```

Controllers In External Files

In larger applications, it is common to store controllers in external files.

Just copy the code between the <script> tags into an external file named [personController.js](#):

AngularJS Example

```
<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>
```

Another Example

For the next example we will create a new controller file:

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

Save the file as [namesController.js](#):

And then use the controller file in an application:

AngularJS Example

```
<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>

```

AngularJS Filters

Filters can be added to expressions and directives using a pipe character.

AngularJS Filters

AngularJS filters can be used to transform data:

Filter	Description
currency	Format a number to a currency format.
filter	Select a subset of items from an array.
lowercase	Format a string to lower case.
orderBy	Orders an array by an expression.
uppercase	Format a string to upper case.

Adding Filters to Expressions

A filter can be added to an expression with a pipe character (|) and a filter.

(For the next two examples we will use the person controller from the previous chapter)

The **uppercase** filter format strings to upper case:

AngularJS Example

```

<div ng-app="myApp" ng-controller="personCtrl">
  <p>The name is {{ lastName | uppercase }}</p>
</div>

```

The **lowercase** filter format strings to lower case:

AngularJS Example

```

<div ng-app="myApp" ng-controller="personCtrl">
  <p>The name is {{ lastName | lowercase }}</p>
</div>

```

The currency Filter

The **currency** filter formats a number as currency:

AngularJS Example

```

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

<input type="number" ng-model="quantity">
<input type="number" ng-model="price">

<p>Total = {{ (quantity * price) | currency }}</p>

</div>

```

Adding Filters to Directives

A filter can be added to a directive with a pipe character (|) and a filter.

The **orderBy** filter orders an array by an expression:

AngularJS Example

```

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

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

</div>

```

Filtering Input

An input filter can be added to a directive with a pipe character (|) and filter followed by a colon and a model name.

The **filter** filter selects a subset of an array:

AngularJS Example

```

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

<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>

```

AngularJS AJAX - \$http

\$http is an AngularJS service for reading data from remote servers.

Providing Data

The following data can be provided by a web server:

<http://www.w3schools.com/angular/customers.php>

```

{
"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

AngularJS **\$http** is a core service for reading data from web servers.

`$http.get(url)` is the function to use for reading server data.

AngularJS Example

```

<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) {
  $http.get("http://www.w3schools.com/angular/customers.php")
    .success(function(response) {$scope.names = response.records;});
});
</script>

```

Application explained:

The AngularJS application is defined by **ng-app**. The application runs inside a `<div>`.

The **ng-controller** directive names the **controller object**.

The **customersCtrl** function is a standard JavaScript **object constructor**.

AngularJS will invoke customersCtrl with a **\$scope** and **\$http** object.

\$scope is the **application object** (the owner of application variables and functions).

\$http is an **XMLHttpRequest object** for requesting external data.

\$http.get() reads **JSON data** from <http://www.w3schools.com/angular/customers.php>.

If **success**, the controller creates a property (**names**) in the scope, with JSON data from the server.

ngularJS Tables

The ng-repeat directive is perfect for displaying tables.

Displaying Data in a Table

Displaying tables with angular is very simple:

AngularJS Example

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

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

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
  $http.get("http://www.w3schools.com/angular/customers.php")
    .success(function (response) {$scope.names = response.records;});
});
</script>
```

Displaying with CSS Style

To make it nice, add some CSS to the page:

CSS Style

```
<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>

```

Display with orderBy Filter

To sort the table, add an **orderBy** filter:

AngularJS Example

```

<table>
  <tr ng-repeat="x in names | orderBy : 'Country'">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

```

Display with uppercase Filter

To display uppercase, add an **uppercase** filter:

AngularJS Example

```

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

```

Display the Table Index (\$index)

To display the table index, add a `<td>` with **\$index**:

AngularJS Example

```

<table>
  <tr ng-repeat="x in names">
    <td>{{ $index + 1 }}</td>
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

```

Using \$even and \$odd

AngularJS Example

```

<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>

```

```
</tr>
</table>
```

AngularJS SQL

The code from the previous chapter can also be used to read from databases.

Fetching Data From a PHP Server Running MySQL

AngularJS Example

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

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

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
  $http.get("http://www.w3schools.com/angular/customers_mysql.php")
    .success(function (response) {$scope.names = response.records;});
});
</script>
```

Fetching Data From an ASP.NET Server Running SQL

AngularJS Example

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

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

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
  $http.get("http://www.w3schools.com/angular/customers_sql.aspx")
    .success(function (response) {$scope.names = response.records;});
});
</script>
```

Server Code Examples

The following section is a listing of the server code used to fetch SQL data.

1. Using PHP and MySQL. Returning JSON.

2. Using PHP and MS Access. Returning JSON.
3. Using ASP.NET, VB, and MS Access. Returning JSON.
4. Using ASP.NET, Razor, and SQL Lite. Returning JSON.

Cross-Site HTTP Requests

Requests for data from a different server (than the requesting page), are called **cross-site** HTTP requests.

Cross-site requests are common on the web. Many pages load CSS, images, and scripts from different servers.

In modern browsers, cross-site HTTP requests **from scripts** are restricted to **same site** for security reasons.

The following line, in our PHP examples, has been added to allow cross-site access.

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

1. Server Code PHP and 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);
?>
```

2. Server Code PHP and 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);
?>

```

3. Server Code ASP.NET, VB and MS Access

```

<%@ Import Namespace="System.IO"%>
<%@ Import Namespace="System.Data"%>
<%@ Import Namespace="System.Data.OleDb"%>
<%
Response.AppendHeader("Access-Control-Allow-Origin", "*")
Response.AppendHeader("Content-type", "application/json")
Dim conn As OleDbConnection
Dim objAdapter As OleDbDataAdapter
Dim objTable As DataTable
Dim objRow As DataRow
Dim objDataSet As New DataSet()
Dim outp
Dim c
conn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;data source=Northwind.mdb")
objAdapter = New OleDbDataAdapter("SELECT CompanyName, City, Country FROM Customers",
conn)
objAdapter.Fill(objDataSet, "myTable")
objTable=objDataSet.Tables("myTable")

outp = ""
c = chr(34)
for each x in objTable.Rows
if outp <> "" then outp = outp & ", "
outp = outp & "{" & c & "Name" & c & ":" & c & x("CompanyName") & c & ", "
outp = outp & c & "City" & c & ":" & c & x("City") & c & ", "
outp = outp & c & "Country" & c & ":" & c & x("Country") & c & "}"
next

outp="{ " & c & "records" & c & ":[ " & outp & "]"
response.write(outp)
conn.close
%>

```

4. Server Code ASP.NET, VB Razor and 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

```

AngularJS HTML DOM

AngularJS has directives for binding application data to the attributes of HTML DOM elements.

The ng-disabled Directive

The **ng-disabled** directive binds AngularJS application data to the disabled attribute of HTML elements.

AngularJS Example

```

<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>

</div>

```

Application explained:

The **ng-disabled** directive binds the application data **mySwitch** to the HTML button's **disabled** attribute.

The **ng-model** directive binds the value of the HTML checkbox element to the value of **mySwitch**.

If the value of **mySwitch** evaluates to **true**, the button will be disabled:

```

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

```

If the value of **mySwitch** evaluates to **false**, the button will not be disabled:

```

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

```

The ng-show Directive

The **ng-show** directive shows or hides an HTML element.

AngularJS Example

```

<div ng-app="">

<p ng-show="true">I am visible.</p>

```

```
<p ng-show="false">I am not visible.</p>
</div>
```

The **ng-show** directive shows (or hides) an HTML element based on the **value** of ng-show.

You can use any expression that evaluates to true or false:

AngularJS Example

```
<div ng-app="">
<p ng-show="hour > 12">I am visible.</p>
</div>
```



In the next chapter, there are more examples, using the click of a button to hide HTML elements.

The ng-hide Directive

The **ng-hide** directive hides or shows an HTML element.

AngularJS Example

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

AngularJS Events

AngularJS has its own HTML events directives.

The ng-click Directive

The **ng-click** directive defines an AngularJS click event.

AngularJS Example

```
<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>
```

Hiding HTML Elements

The **ng-hide** directive can be used to set the **visibility** of a part of an application.

The value **ng-hide="true"** makes an HTML element **invisible**.

The value **ng-hide="false"** makes the element visible.

AngularJS Example

```
<div ng-app="myApp" ng-controller="personCtrl">  
  
<button ng-click="toggle()">Toggle</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>
```

Application explained:

The first part of the **personController** is the same as in the chapter about controllers.

The application has a default property (a variable): **\$scope.myVar = false;**

The **ng-hide** directive sets the visibility, of a `<p>` element with two input fields, according to the value (true or false) of **myVar**.

The function **toggle()** toggles **myVar** between true and false.

The value **ng-hide="true"** makes the element **invisible**.

Showing HTML Elements

The **ng-show** directive can also be used to set the **visibility** of a part of an application.

The value **ng-show="false"** makes an HTML element **invisible**.

The value **ng-show="true"** makes the element visible.

Here is the same example as above, using ng-show instead of ng-hide:

AngularJS Example

```
<div ng-app="myApp" ng-controller="personCtrl">
<button ng-click="toggle()">Toggle</button>

<p ng-show="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 = true;
    $scope.toggle = function() {
        $scope.myVar = !$scope.myVar;
    };
});
</script>
```

AngularJS Modules

An AngularJS module defines an application.

The module is a container for the different parts of an application.

The module is a container for the application controllers.

Controllers always belong to a module.

A Module With One Controller

This application ("myApp") has one controller ("myCtrl"):

AngularJS Example

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

```

Modules and Controllers in Files

It is common in AngularJS applications to put the module and the controllers in JavaScript files.

In this example, "myApp.js" contains an application module definition, while "myCtrl.js" contains the controller:

AngularJS Example

```

<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<body>

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

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

</body>
</html>

```

myApp.js

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



The [] parameter in the module definition can be used to define dependent modules.

myCtrl.js

```

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

```

Functions can Pollute the Global Namespace

Global functions should be avoided in JavaScript. They can easily be overwritten or destroyed by other scripts.

AngularJS modules reduces this problem, by keeping all functions local to the module.

When to Load the Library

While it is common in HTML applications to place scripts at the end of the <body> element, it is recommended that you load the AngularJS library either in the <head> or at the start of the <body>.

This is because calls to angular.module can only be compiled after the library has been loaded.

AngularJS Example

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

<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>
```

AngularJS Forms

An AngularJS form is a collection of input controls.

HTML Controls

HTML input elements are called HTML controls:

- input elements
- select elements
- button elements
- textarea elements

HTML Forms

HTML forms group HTML controls together.

An AngularJS Form Example

First Name:

Last Name:

```
form = {"firstName":"John","lastName":"Doe"}
```

```
master = {"firstName":"John","lastName":"Doe"}
```

Application Code

```
<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>
```



The **novalidate** attribute is new in HTML5. It disables any default browser validation.

Example Explained

The **ng-app** directive defines the AngularJS application.

The **ng-controller** directive defines the application controller.

The **ng-model** directive binds two input elements to the **user** object in the model.

The **formCtrl** function sets initial values to the **master** object, and defines the **reset()** method.

The **reset()** method sets the **user** object equal to the **master** object.

The **ng-click** directive invokes the **reset()** method, only if the button is clicked.

The **novalidate** attribute is not needed for this application, but normally you will use it in AngularJS forms, to override standard HTML5 validation.

AngularJS Input Validation

AngularJS forms and controls can validate input data.

Input Validation

In the previous chapter, you learned about AngularJS forms and controls.

AngularJS forms and controls can provide validation services, and notify users of invalid input.



Client-side validation cannot alone secure user input. Server side validation is also necessary.

Application Code

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



The HTML form attribute **novalidate** is used to disable default browser validation.

Example Explained

The AngularJS directive **ng-model** binds the input elements to the model.

The model object has two properties: **user** and **email**.

Because of **ng-show**, the spans with color:red are displayed only when user or email is **\$dirty** and **\$invalid**.

Property	Description
\$dirty	The user has interacted with the field.
\$valid	The field content is valid.
\$invalid	The field content is invalid.
\$pristine	User has not interacted with the field yet.

AngularJS API

API stands for **A**pplication **P**rogramming **I**nterface.

AngularJS Global API

The AngularJS Global API is a set of global JavaScript functions for performing common tasks like:

- Comparing objects
- Iterating objects
- Converting data

The Global API functions are accessed using the angular object.

Below is a list of some common API functions:

API	Description
angular.lowercase()	Converts a string to lowercase
angular.uppercase()	Converts a string to uppercase
angular.isString()	Returns true if the reference is a string
angular.isNumber()	Returns true if the reference is a number

angular.lowercase()

Example

```
<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>
```

angular.uppercase()

Example

```
<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>
```

angular.isString()

Example

```
<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>
```

angular.isNumber()

Example

```
<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>
```

ngularJS and Twitter Bootstrap

Bootstrap is a popular style sheet. This chapter demonstrates how to use it with AngularJS.

Bootstrap

To include Bootstrap in your AngularJS application, add the following line to the head of your document:

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

If you want to study Bootstrap, visit our [Bootstrap Tutorial](#).

Below is a complete HTML example, with all AngularJS directives and Bootstrap classes explained.

HTML Code

```
<!DOCTYPE html>
<html>
<link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<body ng-app="myApp" ng-controller="userCtrl">

<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>&nbsp;&nbsp;&nbsp;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>

<h3 ng-show="edit">Create New User:</h3>
<h3 ng-hide="edit">Edit User:</h3>

<form class="form-horizontal">
<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>
</form>
```

```

</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>
</form>

<hr>
<button class="btn btn-success" ng-disabled="error || incomplete">
  <span class="glyphicon glyphicon-save"></span> Save Changes
</button>
</div>

<script src = "myUsers.js"></script>
</body>
</html>

```

Directives (Used Above) Explained

AngularJS Directive	Description
<body ng-app	Defines an application for the <body> element
<body ng-controller	Defines a controller for the <body> element
<tr ng-repeat	Repeats the <tr> element for each user in users
<button ng-click	Invoke the function editUser() when the <button> element is clicked
<h3 ng-show	Show the <h3>s element if edit = true
<h3 ng-hide	Hide the <h3> element if edit = true
<input ng-model	Bind the <input> element to the application
<button ng-disabled	Disables the <button> element if error or incomplete = true

Bootstrap Classes Explained

Element	Bootstrap Class	Defines
<div>	container	A content container
<table>	table	A table
<table>	table-striped	A striped table
<button>	btn	A button
<button>	btn-success	A success button
	glyphicon	A glyph icon
	glyphicon-pencil	A pencil icon
	glyphicon-user	A user icon
	glyphicon-save	A save icon
<form>	form-horizontal	A horizontal form
<div>	form-group	A form group
<label>	control-label	A control label
<label>	col-sm-2	A 2 columns span
<div>	col-sm-10	A 10 columns span

JavaScript Code

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;
    }
  };
});

```

JavaScript Code Explained

Scope Properties	Used for
\$scope.fName	Model variable (user first name)
\$scope.lName	Model variable (user last name)
\$scope.passw1	Model variable (user password 1)
\$scope.passw2	Model variable (user password 2)

\$scope.users	Model variable (array of users)
\$scope.edit	Set to true when user clicks on create user.
\$scope.error	Set to true if passw1 not equal passw2
\$scope.incomplete	Set to true if any field is empty (length = 0)
\$scope.editUser	Sets model variables
\$scope.\$watch	Watches model variables
\$scope.test	Tests model variables for errors and incompleteness

AngularJS Includes

With AngularJS, you can include HTML files in HTML.

HTML Includes in Future HTML

Including a portion of HTML in HTML is, unfortunately, not (yet) supported by HTML.

HTML imports is a W3C suggestion <http://www.w3.org> for future versions of HTML:

```
<link rel="import" href="/path/navigation.html">
```

Server Side Includes

Most web servers support Server Side Includes (**SSI**).

With SSI, you can include HTML in HTML before the page is sent to the browser.

PHP Example

```
<?php require("navigation.php"); ?>
```

Client Side Includes

There are many ways to use JavaScript to include HTML in HTML.

The most common way is to use an http request (**AJAX**) to fetch data from a server, and then write the data to the **innerHTML** of an HTML element.

AngularJS Side Includes

With AngularJS, you can include HTML content using the **ng-include** directive:

Example

```
<body>

<div class="container">
  <div ng-include="myUsers_List.htm"></div>
  <div ng-include="myUsers_Form.htm"></div>
</div>

</body>
```

Below is a 4 step introduction.

Step 1: Create the HTML List

myUsers_List.htm

```
<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>&nbsp;&nbsp;&nbsp;Edit
      </button>
    </td>
    <td>{{ user.fName }}</td>
    <td>{{ user.lName }}</td>
  </tr></tbody>
</table>
```

Step 2: Create the HTML Form

myUsers_Form.htm

```
<button class="btn btn-success" ng-click="editUser('new')">
  <span class="glyphicon glyphicon-user"></span> Create New User
</button>
<hr>

<h3 ng-show="edit">Create New User:</h3>
<h3 ng-hide="edit">Edit User:</h3>

<form class="form-horizontal">
<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>
```

```
</div>
</form>
```

```
<hr>
<button class="btn btn-success" ng-disabled="error || incomplete">
  <span class="glyphicon glyphicon-save"></span> Save Changes
</button>
```

Step 3: Create the Controller

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;
  }
});
})
```

Step 4: Create the Main Page

```
<!DOCTYPE html>
<html>
<link rel="stylesheet" href = "http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
<script src= "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

<body ng-app="myApp" ng-controller="userCtrl">

<div class="container">
  <div ng-include="myUsers_List.htm"></div>
  <div ng-include="myUsers_Form.htm"></div>
</div>

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

</body>

</html>
```

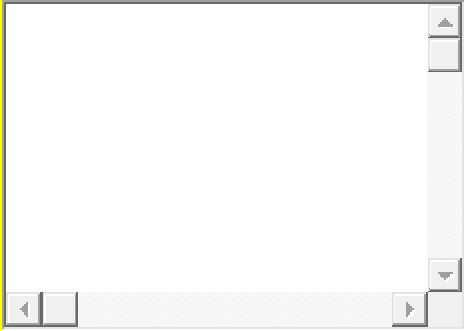
AngularJS Application

It is time to create a real AngularJS Single Page Application (SPA).

An AngularJS Application Example

You have learned more than enough to create your first AngularJS application:

My Note



Number of characters left: **100**

The screenshot shows a web browser window with a yellow background. At the top left, there is a header 'My Note'. Below it is a large, empty text area with a vertical scrollbar on the right and horizontal scrollbars at the bottom. Below the text area, the text 'Number of characters left: 100' is displayed in a bold font.

Application Explained

AngularJS Example

```
<html ng-app="myNoteApp">
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<body>

<div ng-controller="myNoteCtrl">

<h2>My Note</h2>
```

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

```
<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>
```

```
</div>
```

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

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

The application file "myNoteApp.js":

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

The controller file "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");};  
});
```

The `<html>` element is the container of the AngularJS application: `ng-app="myNoteApp"`:

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

A `<div>` in the HTML page is the scope of a controller: `ng-controller="myNoteCtrl"`:

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

An **ng-model** directive binds a `<textarea>` to the controller variable **message**:

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

The two **ng-click** events invoke the controller functions **clear()** and **save()**:

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

An **ng-bind** directive binds the controller function **left()** to a `` displaying the characters left:

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

Your application libraries are added to the page (after the library):

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

AngularJS Application Skeleton

Above you have the skeleton (scaffolding) of a real life AngularJS, single page application (SPA).

The `<html>` element is the "container" for the AngularJS application (**ng-app=**).

A `<div>` elements defines the scope of an AngularJS controller (**ng-controller=**).

You can have many controllers in one application.

An application file (**my...App.js**) defines the application module code.

One or more controller files (**my...Ctrl.js**) defines the controller code.

Summary - How Does it Work?

The ng-app directive is placed at the root element the application.

For single page applications (SPA), the root of the application is often the `<html>` element.

One or more ng-controller directives define the application controllers. Each controller has its own scope: the HTML element where they were defined.

AngularJS starts automatically on the HTML DOMContentLoaded event. If an ng-app directive is found, AngularJS will load any module named in the directive, and compile the DOM with ng-app as the root of the application.

The root of the application can be the whole page, or a smaller portion of the page. The smaller the portion, the faster the application will compile and execute.

AngularJS Examples

Try it Yourself

You can edit the examples online, and click on a button to view the result.

AngularJS Example

```
<div ng-app="">
<p>Name: <input type="text" ng-model="name"></p>
<p>You wrote: {{ name }}</p>
</div>
```

AngularJS Basics

[My first AngularJS Directives](#)

[My first AngularJS Directives \(with valid HTML5\)](#)

[My first AngularJS Expression](#)

[My first AngularJS Controller](#)

[Basic AngularJS Explained](#)

AngularJS Expressions

[A simple Expression](#)
[Expression without ng-app](#)
[Expression with Numbers](#)
[Using ng-bind with Numbers](#)
[Expression with Strings](#)
[Using ng-bind with Strings](#)
[Expression with Objects](#)
[Using ng-bind with Objects](#)
[Expression with Arrays](#)
[Using ng-bind with Arrays](#)

[Expressions Explained](#)

AngularJS Directives

[AngularJS Directives](#)
[The ng-model Directive](#)
[The ng-repeat Directive \(with Arrays\)](#)
[The ng-repeat Directive \(with Objects\)](#)

[Directives Explained](#)

AngularJS Controllers

[AngularJS Controller](#)
[Controller Properties](#)
[Controller Functions](#)
[Controller in JavaScript File I](#)
[Controller in JavaScript File II](#)

[Controllers Explained](#)

AngularJS Filters

[Expression Filter uppercase](#)
[Expression Filter lowercase](#)
[Expression Filter currency](#)
[Directive Filter orderBy](#)
[Input Filters](#)

[Filters Explained](#)

AngularJS XMLHttpRequest

[Reading a static JSON file](#)

[XMLHttpRequest Explained](#)

AngularJS Tables

[Displaying a table \(simple\)](#)
[Displaying a table with CSS](#)
[Displaying a table with an orderBy filter](#)
[Displaying a table with an uppercase filter](#)
[Displaying a table with an index](#)
[Displaying a table with even and odd](#)

[Tables Explained](#)

AngularJS - Reading from SQL Resources

[Reading from a MySQL database](#)

[Reading from a SQL Server database](#)

[Angular SQL Explained](#)

AngularJS HTML DOM

[The ng-disabled Directive](#)

[The ng-show Directive](#)

[The ng-hide Directive](#)

[HTML DOM Explained](#)

AngularJS Events

[The ng-click Directive](#)

[The ng-hide Directive](#)

[The ng-show Directive](#)

[HTML Events Explained](#)

AngularJS Modules

[AngularJS module in body](#)

[AngularJS module in files](#)

[Angular Modules Explained](#)

AngularJS Forms

[AngularJS Forms](#)

[AngularJS Validation](#)

[Angular Forms Explained](#)

AngularJS API

[AngularJS angular.lowercase\(\)](#)

[AngularJS angular.uppercase\(\)](#)

[AngularJS angular.isString\(\)](#)

[AngularJS angular.isNumber\(\)](#)

[API Explained](#)

AngularJS Bootstrap

[AngularJS With Bootstrap](#)

[AngularJS With Bootstrap and Includes](#)

[Bootstrap Explained](#)

AngularJS Applications

[AngularJS Note Application](#)
[AngularJS ToDo Application](#)

[AngularJS Applications](#)

AngularJS Directives

AngularJS directives used in this tutorial:

Directive	Description	Explained
ng-app	Defines the root element of an application.	Directives
ng-bind	Binds the innerHTML of HTML elements to application data.	Introduction
ng-click	Defines the behavior when an element is clicked.	HTML Events
ng-controller	Defines the controller object for an application.	Controllers
ng-disabled	Binds application data to the HTML disabled attribute.	HTML DOM
ng-hide	Hides or shows HTML elements.	HTML DOM
ng-include	Includes HTML in an application.	Includes
ng-init	Defines initial values for an application.	Directives
ng-model	Binds the value of HTML controls to application data.	Directives
ng-repeat	Defines a template for each data in a collection.	Directives
ng-show	Shows or hides HTML elements.	HTML DOM

AngularJS Filters

AngularJS filters used in this tutorial:

Filter	Description
currency	Format a number to a currency format.
filter	Select a subset of items from an array.
lowercase	Format a string to lower case.
orderBy	Orders an array by an expression.
uppercase	Format a string to upper case.

Filters are explained in [Angular Filters](#).

AngularJS Events

AngularJS support the following events:

- ng-click
- ng-dbl-click
- ng-mousedown
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-keydown
- ng-keyup
- ng-keypress
- ng-change

Events are explained in [Angular Events](#).

AngularJS Validation Properties

- \$dirty
- \$invalid
- \$error

Validation is explained in [Angular Validation](#).

AngularJS Global API

Converting

API	Description
angular.lowercase()	Converts a string to lowercase
angular.uppercase()	Converts a string to uppercase
angular.copy()	Creates a deep copy of an object or an array
angular.forEach()	Executes a function for each element in an object or array

Comparing

API	Description
angular.isArray()	Returns true if the reference is an array
angular.isDate()	Returns true if the reference is a date
angular.isDefined()	Returns true if the reference is defined
angular.isElement()	Returns true if the reference is a DOM element
angular.isFunction()	Returns true if the reference is a function
angular.isNumber()	Returns true if the reference is a number
angular.isObject()	Returns true if the reference is an object
angular.isString()	Returns true if the reference is a string
angular.isUndefined()	Returns true if the reference is undefined
angular.equals()	Returns true if two references are equal

JSON

API	Description
angular.fromJSON()	Deserializes a JSON string
angular.toJSON()	Serializes a JSON string

Basic

API	Description
angular.bootstrap()	Starts AngularJS manually
angular.element()	Wraps an HTML element as an jQuery element
angular.module()	Creates, registers, or retrieves an AngularJS module

The Global API is explained in [Angular API](#).