AngularJS Tutorial

Sunnie Chung

http://www.w3schools.com/angular/angular_intro.asp

http://www.w3schools.com/angular/angular_modules.asp

http://www.w3schools.com/angular/angular_http.asp

http://www.w3schools.com/angular/angular_sql.asp

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="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/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="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/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 element is the "owner" of an AngularJS **application**.

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

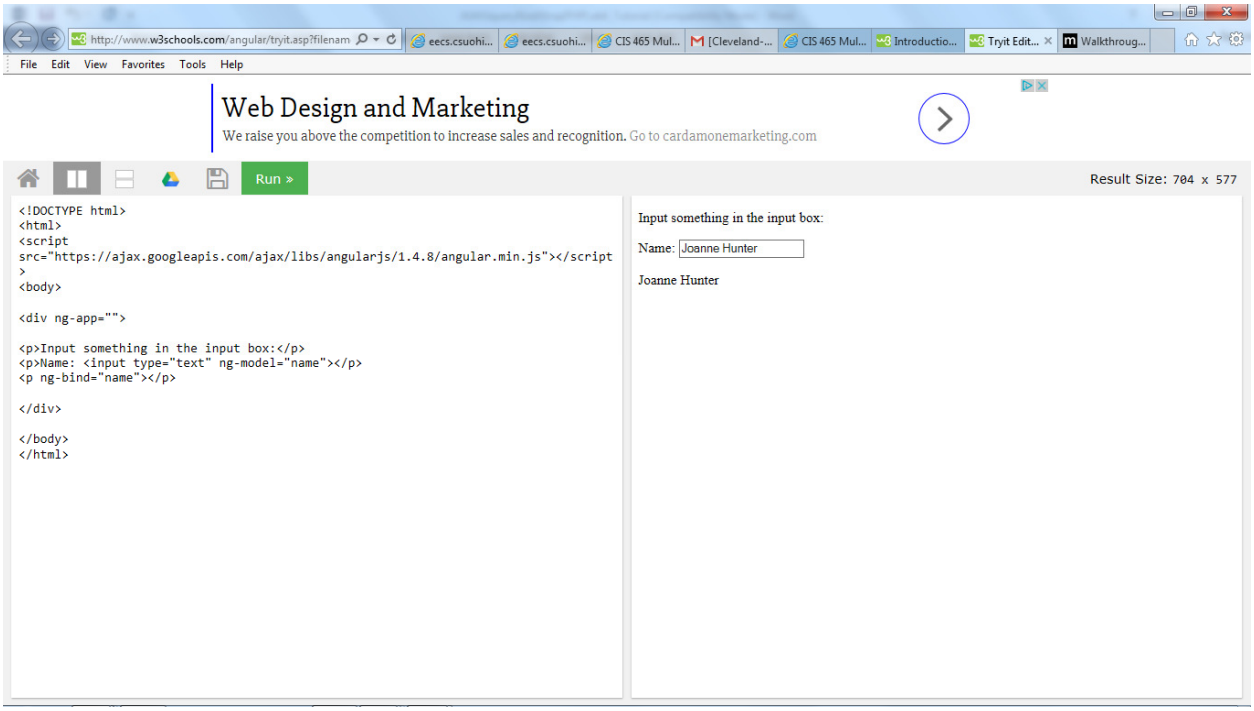
The **ng-bind** directive binds the **innerHTML** of the <p> element to the application variable **name**.

AngularJS Directives

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

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

AngularJS Example



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

AngularJS controllers are regular **JavaScript Objects**.

AngularJS Modules

‹ Previous Next ›

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.

Creating a Module

A module is created by using the AngularJS function `angular.module`

```
<div ng-app="myApp">...</div>

<script>

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

</script>
```

The "myApp" parameter refers to an HTML element in which the application will run.

Now you can add controllers, directives, filters, and more, to your AngularJS application.

Adding a Controller

Add a controller to your application, and refer to the controller with the `ng-controller` directive:

Example

```
<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>
```

You will learn more about controllers later in this tutorial.

Adding a Directive

AngularJS has a set of built-in directives which you can use to add functionality to your application.

For a full reference, visit our AngularJS directive reference.

In addition you can use the module to add your own directives to your applications:

Example

```
<div ng-app="myApp" w3-test-directive></div>

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

app.directive("w3TestDirective", function() {
    return {
        template : "I was made in a directive constructor!"
    };
});
</script>
```

You will learn more about directives later in this tutorial.

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:

Example

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/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.

Without the [] parameter, you are not *creating* a new module, but *retrieving* an existing one.

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.

Example

```
<!DOCTYPE html>
<html>
<body>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/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 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 .

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

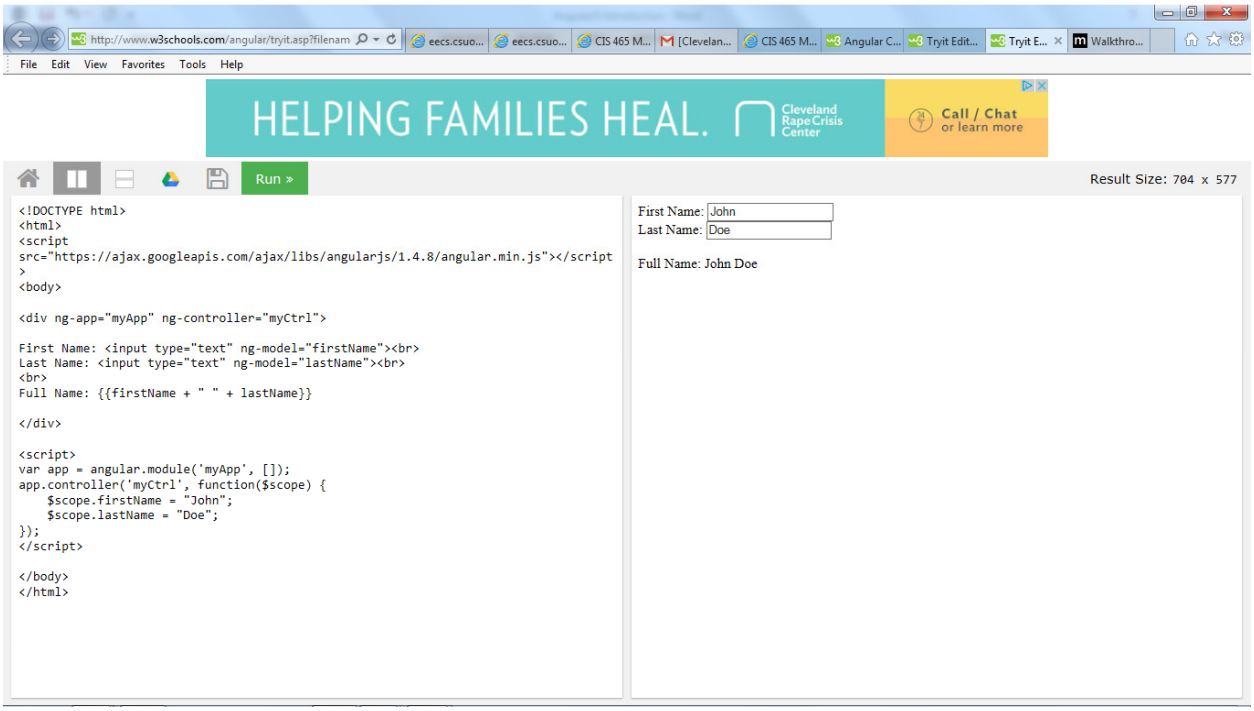
The **myCtrl** function is a JavaScript function.

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: {{fullName()}}

</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 Forms

Forms in AngularJS provides data-binding and validation of input controls.

Input Controls

Input controls are the HTML input elements:

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

Data-Binding

Input controls provides data-binding by using the `ng-model` directive.

```
<input type="text" ng-model="firstname">
```

The application does now have a property named `firstname`.

The `ng-model` directive binds the input controller to the rest of your application.

The property `firstname`, can be referred to in a controller:

Example

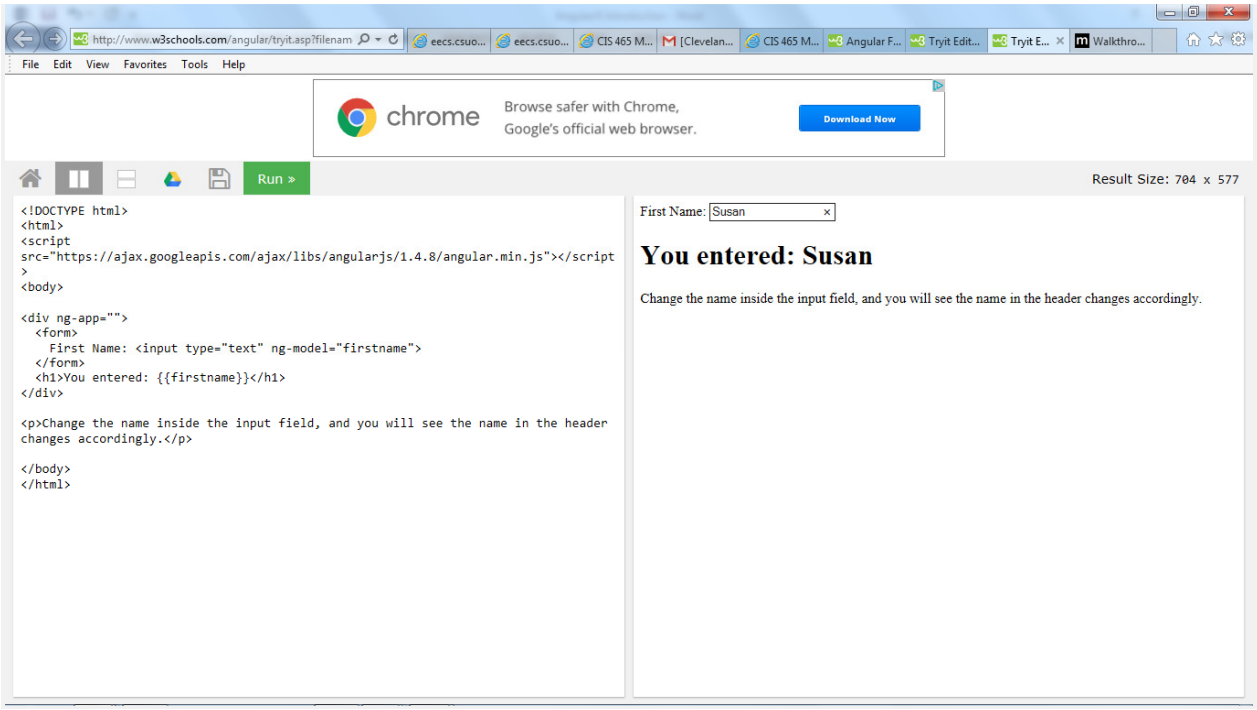
```
<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
    $scope.firstname = "John";
});
</script>
```

It can also be referred to elsewhere in the application:

Example

```
<form>
First Name: <input type="text" ng-model="firstname">
</form>

<h1>You entered: {{firstname}}</h1>
```



Checkbox

A checkbox has the value `true` or `false`. Apply the `ng-model` directive to a checkbox, and use its value in your application.

Example

Show the header if the checkbox is checked:

```
<form>
    Check to show a header:
    <input type="checkbox" ng-model="myVar">
```

```
</form>

<h1 ng-show="myVar">My Header</h1>
```

Radiobuttons

Bind radio buttons to your application with the `ng-model` directive.

Radio buttons with the same `ng-model` can have different values, but only the selected one will be used.

Example

Display some text, based on the value of the selected radio button:

```
 <form>
Pick a topic:
<input type="radio" ng-model="myVar" value="dogs">Dogs
<input type="radio" ng-model="myVar" value="tuts">Tutorials
<input type="radio" ng-model="myVar" value="cars">Cars
</form>
```

The value of myVar will be either `dogs`, `tuts`, or `cars`.

Selectbox

Bind select boxes to your application with the `ng-model` directive.

The property defined in the `ng-model` attribute will have the value of the selected option in the selectbox.

Example

Display some text, based on the value of the selected option:

```
<form>
Select a topic:
<select ng-model="myVar">
    <option value="">
    <option value="dogs">Dogs
    <option value="tuts">Tutorials
    <option value="cars">Cars
</select>
</form>
```

The value of myVar will be either `dogs`, `tuts`, or `cars`.

An AngularJS Form Example

First Name:

Last Name:

RESET

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.

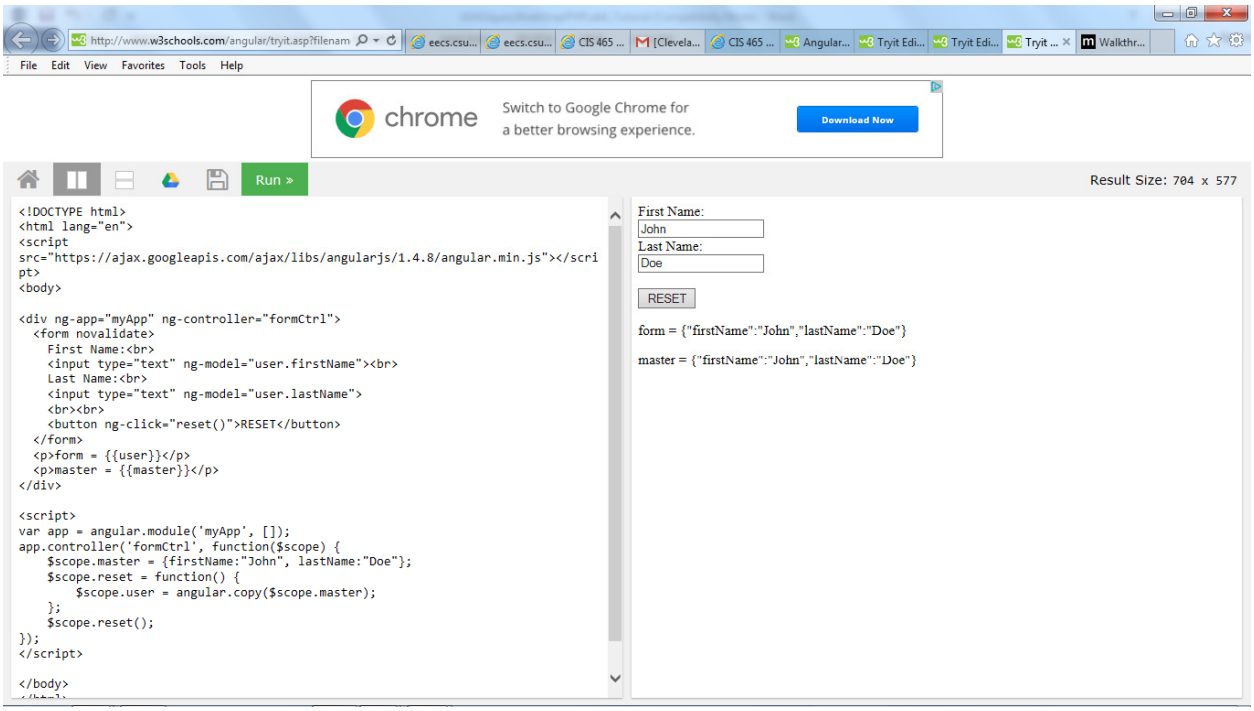
The **ng-model** directive binds two input elements to the **user** object in the model.

The **formCtrl** controller 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 AJAX - $http

http://www.w3schools.com/angular/angular_http.asp

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

AngularJS $http

The AngularJS `$http` service makes a request to the server, and returns a response.

Example

Make a simple request to the server, and display the result in a header:

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

<p>Today's welcome message is:</p>
```

```
<h1>{{myWelcome}}</h1>

</div>

<script>

var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http.get("welcome.htm")
    .then(function(response) {
        $scope.myWelcome = response.data;
    });
});

</script>
```

Methods

The example above uses the `.get` method of the `$http` service.

The .get method is a shortcut method of the $http service. There are several shortcut methods:

- `.delete()`
- `.get()`
- `.head()`
- `.jsonp()`
- `.patch()`
- `.post()`
- `.put()`

The methods above are all shortcuts of calling the $http service:

Example

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http({
        method : "GET",
        url : "welcome.htm"
    }).then(function mySucces(response) {
```

```
        $scope.myWelcome = response.data;
    }, function myError(response) {
        $scope.myWelcome = response.statusText;
    });
});
```

The example above executes the $http service with an object as an argument. The object is specifying the HTTP method, the url, what to do on success, and what to do on failure.

Properties

The response from the server is an object with these properties:

- `.config` the object used to generate the request.
- `.data` a string, or an object, carrying the response from the server.
- `.headers` a function to use to get header information.
- `.status` a number defining the HTTP status.
- `.statusText` a string defining the HTTP status.

Example

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http.get("welcome.htm")
    .then(function(response) {
        $scope.content = response.data;
        $scope.statuscode = response.status;
        $scope.statustext = response.statustext;
    });
});
```

To handle errors, add one more functions to the `.then` method:

Example

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http.get("wrongfilename.htm")
```

```
    .then(function(response) {
        //First function handles success
        $scope.content = response.data;
    }, function(response) {
        //Second function handles error
        $scope.content = "Something went wrong";
    });
});
```

JSON

The data you get from the response is expected to be in JSON format.

JSON is a great way of transporting data, and it is easy to use within AngularJS, or any other JavaScript.

Example: On the server we have a file that returns a JSON object containing 15 customers, all wrapped in array called `records`.

Take a look at the JSON object.

✖

customers.php


```
{
  "records": [
    {
      "Name": "Alfreds Futterkiste",
      "City": "Berlin",
      "Country": "Germany"
    },
    {
      "Name": "Ana Trujillo Emparedados y helados",
      "City": "México D.F.",
      "Country": "Mexico"
    },
    {
      "Name": "Antonio Moreno Taquería",
      "City": "México D.F.",
      "Country": "Mexico"
```

```
},
{
  "Name": "Around the Horn",
  "City": "London",
  "Country": "UK"
},
{
  "Name": "B's Beverages",
  "City": "London",
  "Country": "UK"
},
{
  "Name": "Berglunds snabbköp",
  "City": "Luleå",
  "Country": "Sweden"
},
{
  "Name": "Blauer See Delikatessen",
  "City": "Mannheim",
  "Country": "Germany"
},
{
  "Name": "Blondel père et fils",
  "City": "Strasbourg",
  "Country": "France"
},
{
  "Name": "Bólido Comidas preparadas",
  "City": "Madrid",
  "Country": "Spain"
},
{
  "Name": "Bon app'",
  "City": "Marseille",
  "Country": "France"
},
{
  "Name": "Bottom-Dollar Marketse",
  "City": "Tsawassen",
  "Country": "Canada"
},
{
  "Name": "Cactus Comidas para llevar",
  "City": "Buenos Aires",
  "Country": "Argentina"
},
{
  "Name": "Centro comercial Moctezuma",
  "City": "México D.F.",
```

```
      "Country": "Mexico"
    },
    {
      "Name": "Chop-suey Chinese",
      "City": "Bern",
      "Country": "Switzerland"
    },
    {
      "Name": "Comércio Mineiro",
      "City": "São Paulo",
      "Country": "Brazil"
    }
  ]
}
```

Example

The ng-repeat directive is perfect for looping through an array:

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

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

</div>

<script>

var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
    $http.get("customers.php").then(function(response) {
        $scope.myData = response.data.records;
    });
});

</script>
```

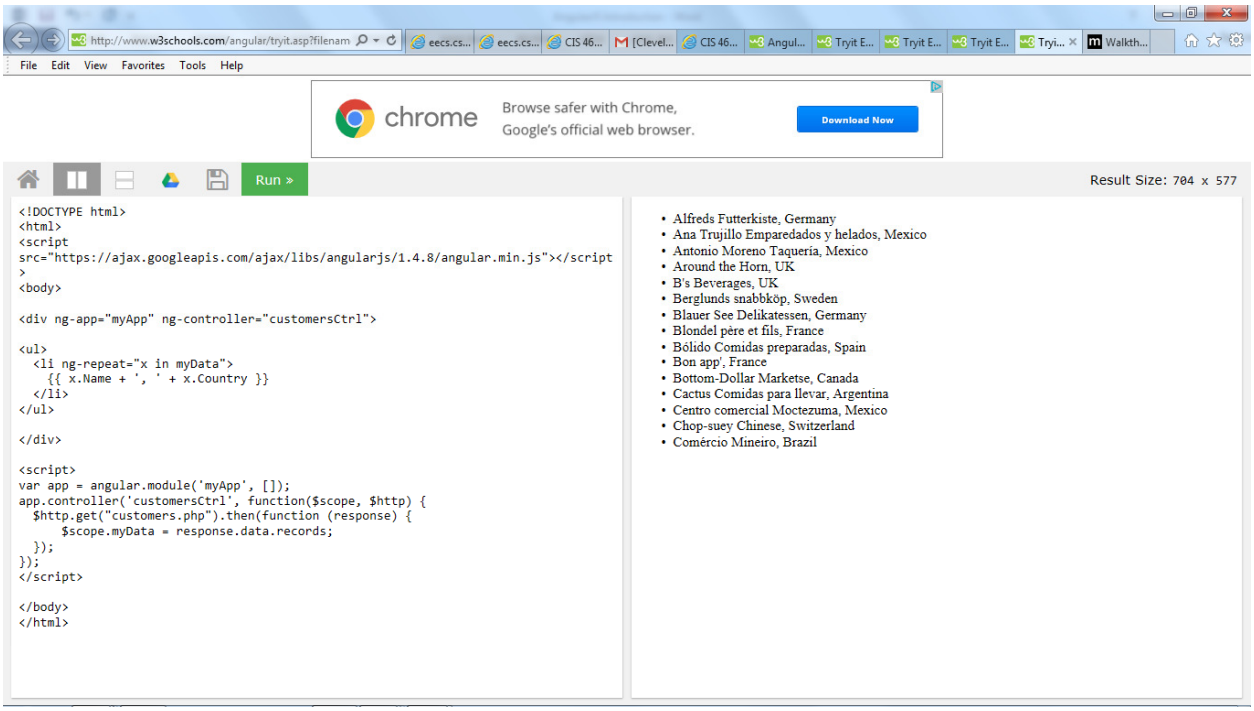
Application explained:

The application defines the `customersCtrl` controller, with a `$scope` and `$http` object.

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

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

On success, the controller creates a property, `myData`, in the scope, with JSON data from the server.



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) {
    $http.get("http://www.w3schools.com/angular/customers_mysql.php")
    .then(function (response) {$scope.names = response.data.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) {
    $http.get("http://www.w3schools.com/angular/customers_sql.aspx")
    .then(function (response) {$scope.names = response.data.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: *");
```


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


---

1. Server Code PHP and MySQL

```php
<?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
<?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, Razor C# 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
```