

Hands-On With Angularjs

Using ASP.NET

This free book is provided by courtesy of C# Corner and Mindcracker Network and its authors. Feel free to share this book with your friends and co-workers. Please do not reproduce, republish, edit or copy this book.

ANUBHAV CHAUDHARY

INDEX

Page no

Hello World Using Angularjs in ASP.Net Application	1- 3
Various Ways To Initialize AngularJS	4-7
Controller in AngularJS	8-11
Passing Argument in the Controller of AngularJS	12-16
Controller Method of Angular Module in AngularJS	17-21
Scope Inheritance in Controller Method of AngularJS	22-25
Using Ng-init and Ng-repeat Directive of AngularJS in ASP.Net Application	26-28
Filter Data in ASP.Net Application Using AngularJS	29-32
Alert Box in AngularJS	33-36
ng-change Directive of AngularJS	37-40
Ng-click and Ng-blur Directives in AngularJS	41-44
Required Field and Email Validation Using AngularJS	45-49
Minimunm Length and Maximum Length Validation Using AngularJS	50-54
Custom Validation Using AngularJS	55-58
ng-BindHtml Directive of AngularJS	59-62
Apply Validation For Checking Valid URL Using AngularJS	63-66
Bind Multiple Values Using AngularJS	67-70
Change Style Dynamically in Various Ways Using AngularJS	71-76
Determine Whether User Has Used Cut, Copy or Paste For Data Using AngularJS	77-82
ng-disabled Directive of AngularJS	83-86
ngClassOdd and ngClassEven Directives of AngularJS	87-91

ng-show Directive of AngularJS	92-95
ng-hide Directive of AngularJS	96-99
ng-dblclick Directive of AngularJS	100-104
Key Up and Key Down Event in AngularJS	105-108
ng-Switch Directive of AngularJS	109-112
ng-Transclude Directive of AngularJS	113-116
ng-Value Directive of AngularJS	117-119
Convert Text To Uppercase Using AngularJS	120-122
Apply Sorting Using AngularJS	123-128
Convert Text To Lowercase Letters Using AngularJS	129-132
Apply "Go to Bottom" and "Go Up" on Click of Anchors	133-137
Logging in Browser Console Using the AngularJS	138-143
\$timeout in AngularJS	144-146
\$exceptionHandler in AngularJS	147-149
\$window in AngularJS	150-153
\$rootScope Service in AngularJS	154-156
Enter and Leave Animation Using AngularJS	157-162
Add Animation To Your Application Using AngularJS	163-166
\$rootElement Service in AngularJS	167-170
Fade In and Fade Out Animation Using AngularJS	171-175
Use ng-class With CSS3 Transition in AngularJS	176-179
Include Multiple Partial Templates in Single Application Using Ng-Template...	180-184
Compare Two Values Using angular.equals	185-188
Ng-Include Directive of AngularJS	189-193

Create To Do List Using AngularJS	194-199
Archive Completed Work From To Do List Using AngularJS	200-206
Show Data in Dynamic Grid Using AngularJS	207-211
Apply CRUD Operations in Dynamic Grid Using AngularJS	212-217
Using angular.forEach API in AngularJS	218-222
Create File Uploader Using AngularJS	223-226
Create Multi File Uploader Using AngularJS	227-230
Use \$filter Service in AngularJS	231-234
.directive and .filter Service in AngularJS	235-238
\$httpBackend Service in AngularJS	239-242
Sanitize Data Using AngularJS	243-246
\$cookies Service in AngularJS	247-250
Cookies in AngularJS	251-253
\$interval Service Provided by AngularJS	254-256
NgInvalid in AngularJS	257-260
NgDirty in AngularJS	261-264
Ternary Operators in AngularJS	265-268
Make AJAX Call and Return JSON Using AngularJS	269-271
Call \$scope, Model of Child Page on Parent Page in AngularJS	272-274
Directive to Allow Decimal Numbers With a Single Decimal Point (.) Using AngularJS	275-279
Directive to Allow Only Numbers Using AngularJS	280-283
Call Function, \$scope, Model of One Page From/on Second Page in AngularJS	284-288
Send Object of Objects From AngularJS to WebAPI	289-295
Get Attributes of Element in AngularJS Just Like jQuery	296-298

Set and Clear Timeout Using \$timeout in AngularJS	299-301
Show Data in Grid Format Using AngularJS and WEB API	302-307
Retain Focus On Invalid Field Using AngularJS Custom Directive	308-312
MVC Routing With AngularJS Routing	313-319
Conditional Routing In AngularJS	320-326
Promise In AngularJS	327-332
Different Kinds Of Scopes In Custom Directives	333-338
Isolated Scopes In Custom Directives	339-348
Use Multiple Modules On Same Page In AngularJS	349-351
\$templateCache In AngularJS	352-354

About The Author

Anubhav Chaudhary is a Software Developer whose rich skill set includes working with ASP.NET, jQuery, LINQ, AngularJS, MY SQL, SQL Server, MVC, etc.



ANUBHAV CHAUDHARY
(Software Developer)

Hello World Using Angularjs in ASP.Net Application

Introduction

This shows how to create a Hello World example using Angularjs in an ASP.NET application.

Angularjs works great on HTML, MVC and ASP.NET as well, it contains some important directives like np-model, np-bind, np-app, np-controller, all these will be explained with examples for them. Let's create a Simple Hello World Example that is first step of learning any new technology, framework and so on.

Step 1 - First of all you need to add an external Angular.js file to your application, for this you can go to the AngularJS official site.

After downloading the external file you now need to add this file to the Head section of your application.

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

Step 2 - Now I will create a JavaScript function that will work for showing the Hello World. Add this JavaScript function in the head section of your application.

```
<script>
function HelloWorld($scope) {
    $scope.test = 'World';
}
</script>
```

Here I had created a function named as "HelloWorld", in this HelloWorld "\$scope" makes a connection between the function and the View, in other words the Design part. "Test" is a variable that will be holding "World" as it's default value.

Step 3 - Now I will create a Div, TextBox and some Labels for showing the Hello World example.

```
<body>
<div ng-controller="HelloWorld">
  Your Name: <input type="text" ng-model="test"/>
  <hr/>
  Hello {{test || "World"}}!
</div>
</body>
```

Here in the Div you can see that a directive is used named "ng-controller" that consists of the name of the JavaScript function that was applied to a specific Div, Span and so on.

Then an input tag is used in which "ng-model" provides the Binding between the View and the Model. In ng-model it has "test" that means the value entered in this TextBox will replace the default value provided in the JavaScript Function.

One more thing is to be done and that is to add "ng-app" in the HTML tag.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

ng-app declares that it is the AngularJS page, if you don't add this directive in the HTML tag then your application will not work properly because it will become unable to detect the other directives of Angular.

Now our Hello World application is ready to be executed.

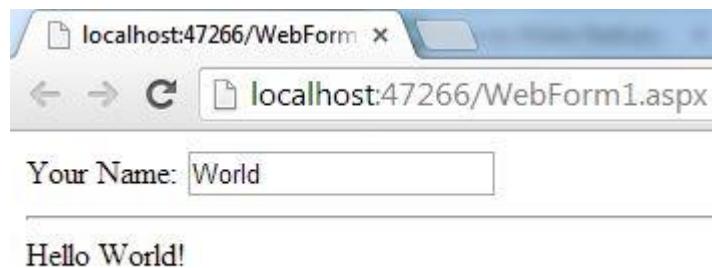
The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head>
  <script src="angular.min.js"></script>
<script>
  function HelloWorld($scope) {
    $scope.test = 'World';
  }
</script>
</head>
<body>
<div ng-controller="HelloWorld">
  Your Name: <input type="text" ng-model="test"/>
```

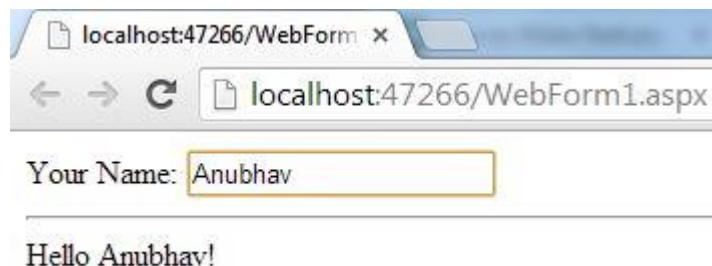
```
<hr/>
Hello {{test || "World"}}!
</div>
</body>
</html>
```

Output:

On running the application you will see the default value that was provided in the function, "Hello World".



Now if I start making changes in the TextBox then the changes will be instantly seen in the Label.



Various Ways To Initialize AngularJS

Introduction

In this I provide three ways to initialize AngularJS in your application.

There are three ways to initialize AngularJS in your application, the first one is Automatic Initialization and is the simplest one that you saw in the previous article, the second one is also Automatic Initialization but is a little difficult from the first one, the third is the most difficult and is done manually.

In all the methods, one thing is common; that is that the external JS file must be added, so the first thing you need to do is to add the AngularJS file in the Head Section of the application. You can download it from the Angular Website.

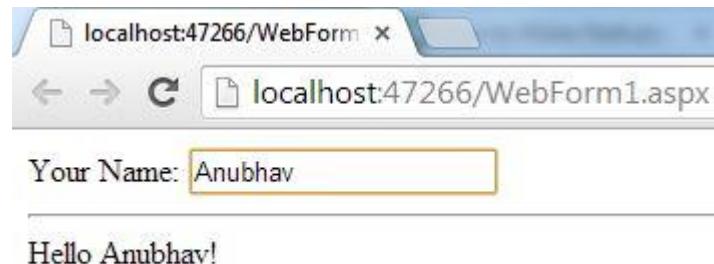
First Method-

In the first method a simple ng-app is written in the HTML tag and the controller is defined as the function name, for example:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script src="angular.min.js"></script>
  <script>
    function HelloWorld($scope) {
      $scope.test = 'World';
    }
  </script>
  </head>
  <body>
    <div ng-controller="HelloWorld">
      Your Name: <input type="text" ng-model="test"/>
      <hr/>
      Hello {{test || "World"}}!
    </div>
  </body>
</html>
```

You can see that in the HTML tag ng-app is simply written, no value is assigned to it, the function name is written as HelloWorld that is later defined as the Controller name.

It's **output** will be as follows:



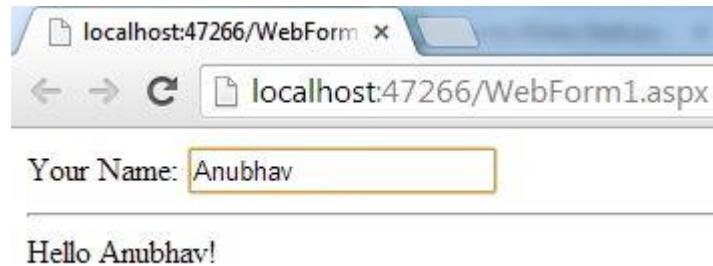
Second Method-

In the second method the ng-app value is defined in the angular.module defined in a JavaScript Tag, the Controller name is also defined in the Controller() and not as a function name. For example:

```
<html ng-app="demo" xmlns="http://www.w3.org/1999/xhtml">
<head>
<script src="angular.min.js"></script>
<script>
angular.module('demo', []).controller('HelloWorld', function ($scope) {
  $scope.test = 'World';
});
</script>
</head>
<body>
<div ng-controller="HelloWorld">
  Your Name: <input type="text" ng-model="test"/>
  <hr/>
  Hello {{test || "World"}}!
</div>
</body>
</html>
```

Here in the Script tag you can see that the angular.module consists of "demo" that is the value for ng-app, Then "HelloWorld" is provided in the Controller() that is the value for ng-controller, the function is created within the controller but the variable test is defined as it was defined previously.

It's **output** will be as follows:



Third Method-

Normally only one AngularJS application can be initialized per HTML document but if we want to initialize more AngularJS applications in a single document then we need to use the Bootstrap method and initialize them manually. The third method is a manual method, this method is used in the case where we want more control over the initialization process. In this we use the Bootstrap Method. The an example of this type of method is:

```
<html ng-app="demo" xmlns="http://www.w3.org/1999/xhtml">
<head>
<script src="angular.min.js"></script>
<script>
    angular.element(document).ready(function() {
        angular.module('demo', []).controller('HelloWorld', function ($scope) {
            $scope.test = 'World');
            angular.bootstrap(document, ['demo']);
        });
    });
</script>
</head>
<body>
<div ng-controller="HelloWorld">
    Your Name: <input type="text" ng-model="test"/>
    <hr/>
    Hello {{test || "World"}}!
</div>

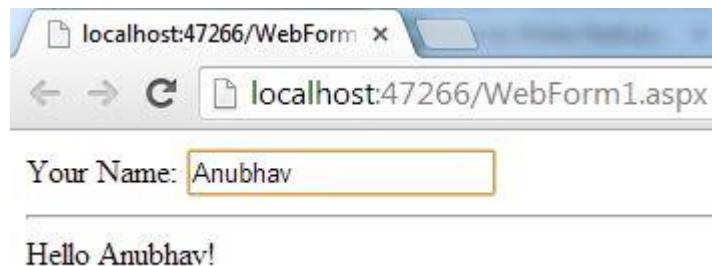
```

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

Here first the ready function is created since it is created in a normal JavaScript program.

Then similar things are done that were done in the second method, but the noticeable thing is that the name of the module is defined at the second place in the bootstrap.

It's **output** will be as follows:



You can see that similar output is obtained in all the Methods.

Controller in AngularJS

Introduction

This explains the Controller in AngularJS. In AngularJS "Controller" has a different meaning, here the JavaScript Constructor function is referred to as a "Constructor" to augment the Angular Scope.

The Controller is called in DOM using the ng-controller directive, the Controller defines the initial state of the \$scope object.

Now I will show you an example by which it will become easy for you to understand the Controller in AngularJS.

Step 1 - First of all you need to add AngularJS to your application, for this you need to download it's external .js file that can be downloaded from the AngularJS Official Website.

Now add this external file in the Head Section of your application:

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

Step 2 - The following is the ViewModel or script part of this application:

```
<script>
function ObservingCar($scope) {
    $scope.cartest = 'Awesome';
    $scope.test1 = function () {
        $scope.cartest = 'Good';
    }
    $scope.test2 = function () {
        $scope.cartest = 'Not Good';
    }
}
</script>
```

Here ObservingCar is the Controller of this application, in this Controller the initial state of \$scope is defined as "Awesome" that is defined in cartest.

Then two functions are created using \$scope, the first function will change the value of cartest to "Good" and the second function will change the value of cartest to "NotGood".

Either of these functions will be called on the click of a button. Now the work on ViewModel is completed, so we just need to work on the View part.

Step 3 - Now I will work on the View part or the design part of this application. I'll add two buttons and a few Labels that will help us to show the demonstration in an easy way.

```
<body>
<div ng-controller="ObservingCar">
  <h2>How Do You Like The Car??</h2>
  <h3>This Car is {{cartest}}</h3>
  <button ng-click="test1()">Good</button>
  <button ng-click="test2()">Not Good</button>
</div>
</body>
```

Now our application is created and is ready for execution. The complete code of this application is as follows:

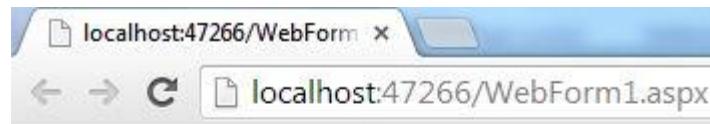
```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head>
<script src="angular.min.js"></script>
<script>
  function ObservingCar($scope) {
    $scope.cartest = 'Awesome';
    $scope.test1 = function () {
      $scope.cartest = 'Good';
    }
    $scope.test2 = function () {
      $scope.cartest = 'Not Good';
    }
  }
</script>
</head>
```

```
<body>
<div ng-controller="ObservingCar">
  <h2>How Do You Like The Car??</h2>
  <h3>This Car is {{cartest}}</h3>
  <button ng-click="test1()">Good</button>
  <button ng-click="test2()">Not Good</button>
</div>
</body>
</html>
```

Output: On running the application you will get output like this:



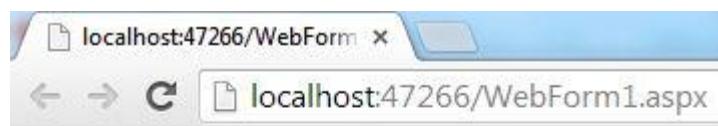
You can see that currently the initial value of \$scope is shown in the output that was "awesome", but as I click on one of the buttons the value of \$scope is updated and is reflected in the output.



How Do You Like The Car??

This Car is Good

Now I again click on the "Next" button that was bound to other function of \$scope and again the value of \$scope is updated.



How Do You Like The Car??

This Car is Not Good

Passing Argument in the Controller of AngularJS

Introduction

In this I will show you how to pass an argument in the Controller of AngularJS.

Let's create an application that will help you understand how arguments are passed in the Controller of AngularJS.

Step 1 - First of all you need to add AngularJS to your application, for this you need to download it's external .js file that can be downloaded from the AngularJS official website.

Now add this external file in the Head section of your application as in the following:

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

Step 2 - Now I will work on the ViewModel or Script part of this application.

```
<script>
function carcontrol($scope) {
    $scope.notgood = 'Not so Good';
    $scope.cartest = 'Awesome';
    $scope.test1 = function (cartest) {
        $scope.cartest = cartest;
    }
};
</script>
```

Here carcontrol is the controller of this application, in this Controller the initial state of \$scope is defined as "Awesome" defined in cartest.

"Not so Good" can also be termed as an initial state but it is used as the initial state in the TextBox that you will see later in this article.

Then a function is created named test1, this is taking cartest as it's argument.

Now the work on ViewModel is completed, now we just need to work on the View part.

Step 3 - Now I will work on the View part or the design part of this application, I'll add two buttons and a few Labels that will help us to show the Demonstration in an easy way.

```
<body>
  <div ng-controller="carcontrol">
    <input ng-model="notgood" />
    <h2>How Do You Like The Car??</h2>
    <h3>This Car is {{cartest}}</h3>
    <button ng-click="test1('Really Good')">Good</button>
    <button ng-click="test1(notgood)">Not Good</button>
  </div>
</body>
```

Here a div is taken in which the controller is called, in this Div a TextBox is taken to which "notgood" is bound using the ng-model. As I told you in Step 2, an initial value is passed in the notgood, so our TextBox will show this initial value on running the page.

After this cartest is bound to the H3, that will help us to show what selection is made by the user because the value of this H3 tag will change depending on the change of selection made by the user.

Then two buttons are used, in the first button a static value is passed when the binding is done to the test1, so a click of this button will pass this static value to the H3 tag.

Now our application is created and can be run to check how it's working.

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script src="angular.min.js"></script>
  <script>
    function carcontrol($scope) {
      $scope.notgood = 'Not so Good';
      $scope.cartest = 'Awesome';
      $scope.test1 = function (cartest) {
        $scope.cartest = cartest;
```

```

        }
    };
</script>
</head>
<body>
<div ng-controller="carcontrol">
    <input ng-model="notgood" />
    <h2>How Do You Like The Car??</h2>
    <h3>This Car is {{cartest}}</h3>
    <button ng-click="test1('Really Good')">Good</button>
    <button ng-click="test1(notgood)">Not Good</button>
</div>
</body>
</html>

```

Output: On running the application the default values will be seen by the user.



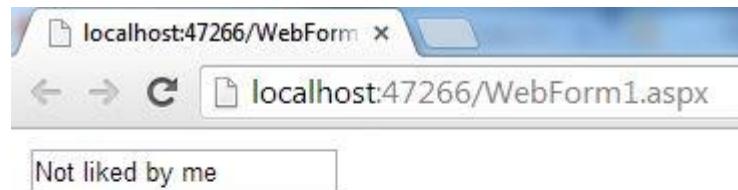
Now if the user clicks on the first button then the value of H3 will change depending on the value bound to this button.



Now If I click on the second button then whatever value is passed through the TextBox will be shown in the H3.



Now I am changing the value in the TextBox, you can see that the value of H3 is changed accordingly.



How Do You Like The Car??

This Car is Not liked by me

Controller Method of Angular Module in AngularJS

Introduction

In previous we used a Controller function in the Global Scope, AngularJS allows us to use a Controller in Global Scope but it is not recommended (by the AngularJS official website), instead we are asked to use the Controller method of the Angular Module.

This topic will help you to understand how to use a Controller as a method of an Angular Module in AngularJS. I will work on a similar application in which we had worked on previously but this time a few things will be changed because this time we will not use the Controller in Global Scope.

Step 1 - First of all you need to add AngularJS to your application, for this you need to download its external .js file that can be downloaded from the AngularJS official website, or you can download the zip file provided at the top of this article or you can click on this link: [AngularJS](#).

Now add this external file in the Head Section of your application as in the following:

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

Step 2 - Now I will work on the ViewModel or Script part of this application.

```
<script>
  var mod = angular.module('testmod', []);
  mod.controller('carcontrol',['$scope',function ($scope) {
    $scope.notgood = 'Not so Good';
    $scope.cartest = 'Awesome';
    $scope.test1 = function (cartest) {
      $scope.cartest = cartest;
    }
  }]);
</script>
```

Here you can see that I created an object of the Angular module that will be accessed using "testmod", you can rename this module depending on your preferences.

Then the Controller method of the Angular Module is called whose name is written as "carcontrol". I had created a Controller function where a few initial values are provided that can be changed at run time. An argument is also passed in the controller as "cartest".

Step 3 - Now I will work on the View part or the design part of this application, I'll add two buttons and a few Labels that will help us to show the Demonstration in an easy way.

```
<body>
  <div ng-controller="carcontrol">
    <input ng-model="notgood" />
    <h2>How Do You Like The Car??</h2>
    <h3>This Car is {{cartest}}</h3>
    <button ng-click="test1('Really Good')">Good</button>
    <button ng-click="test1(notgood)">Not Good</button>
  </div>
</body>
```

Here Controller is assigned to the Div using the ng-controller. After this an initial value is passed to the Textbox in ng-model that will allow us to change this value at run time.

After this cartest is bound to the H3, that will help us to show what selection is made by the user because the value of this H3 tag will change depending on the change of selection made by the user.

Then two buttons are used, in the first button a static value is passed when the binding is done to the test1, so a click of this button will pass this static value to the H3 tag.

Now our application is created and can be run to check how it's working.

The complete code of this application is as follows:

```
<html ng-app="testmod" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script src="angular.min.js"></script>
  <script>
```

```

var mod = angular.module('testmod', []);
mod.controller('carcontrol',['$scope',function ($scope) {
    $scope.notgood = 'Not so Good';
    $scope.cartest = 'Awesome';
    $scope.test1 = function (cartest) {
        $scope.cartest = cartest;
    }
}]);
</script>
</head>
<body>
<div ng-controller="carcontrol">
<input ng-model="notgood" />
<h2>How Do You Like The Car??</h2>
<h3>This Car is {{cartest}}</h3>
<button ng-click="test1('Really Good')">Good</button>
<button ng-click="test1(notgood)">Not Good</button>
</div>
</body>
</html>

```

Output: On running the application the default values will be seen by the user.



Now if the user clicks on the first button then value of H3 will change depending on the value bound to this button.



Now If I click on the second button then whatever value is passed through the Textbox will be shown in the H3.



Now I am changing the value in the Textbox, you can see that the value of H3 is changed accordingly.



Scope Inheritance in Controller Method of AngularJS

Introduction

This explains Scope Inheritance in a Controller Method of AngularJS.

Controllers can be assigned to any number of DOM hierarchies and we know that Controllers are assigned using the ng-controller. Now what ng-controller does is it creates a new child scope each time it is used so in this way a hierarchy of scopes is created that can be inherited from each other. Here parent-child relationships exist, since each child can access its Parent's Properties and Methods so in a similar manner each \$scope received by the Controller can access the Properties and Methods that are defined in it's parent or Grand Parent Controller.

Now I will create an example from which you can easily understand how this concept is working.

Step 1 - This code will be written in the Head Section and in the Script Tag:

```
<script>
  var mod = angular.module('testmod', []);
  mod.controller('maincarcontrol', ['$scope', function ($scope) {
    $scope.name = 'Anubhav';
    $scope.cartest = 'Awesome';
  }]);
  mod.controller('childcarcontrol', ['$scope', function ($scope) {
    $scope.name = 'Mohit';
  }]);
  mod.controller('grandchildcarcontrol', ['$scope', function ($scope) {
    $scope.name = 'Anshika';
    $scope.cartest = 'perfect';
  }]);
</script>
```

Here first I have created an object of angular.module as mod, then I created the first controller method of this Angular Module and named it as "maincarcontrol", in this controller some initial values are assigned to the \$scope.

Then the second controller method is created named "childcarcontrol", in this controller only one initial value is assigned in other words value is assigned to name only and not to cartest.

At the end one more controller is created and named as "grandchildcarcontrol", here again two initial values are defined that are totally different from the first and second.

Step 2 - Now we will work on the View part of this application.

Write this code in the Body section:

```
<body>
  <div id="maindiv" class="main">
    <div ng-controller="maincarcontrol">
      <p>{{name}} says that control of this car is {{cartest}}</p>
      <div ng-controller="childcarcontrol">
        <p>{{name}} says that control of this car is {{cartest}}</p>
        <div ng-controller="grandchildcarcontrol">
          <p>{{name}} says that control of this car is {{cartest}}</p>
        </div>
      </div>
    </div>
  </div>
</body>
```

Here first a main div is created named "maindiv". In this main div three divs are also created that are created in one another and a different controller is assigned to each Div.

In all three divs you can see that a different controller is assigned but similar coding is provided, you will see that the similar code gives different results for all the divs.

Now our application is created and is ready to be executed.

Complete Code of this application is as follows:

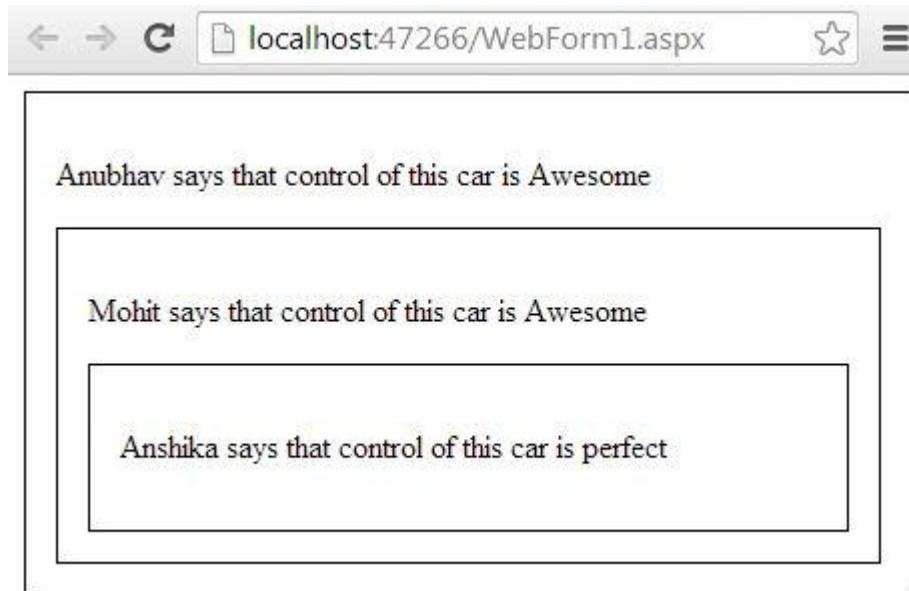
```
<html ng-app="testmod" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script src="angular.min.js"></script>
  <script>
    var mod = angular.module('testmod', []);

```

```

mod.controller('maincarcontrol',['$scope',function ($scope) {
    $scope.name = 'Anubhav';
    $scope.cartest = 'Awesome';
}]);
mod.controller('childcarcontrol', ['$scope', function ($scope)
{
    $scope.name = 'Mohit';
}]);
mod.controller('grandchildcarcontrol', ['$scope', function ($scope)
{
    $scope.name = 'Anshika';
    $scope.cartest = 'perfect';
}
]);
</script>
<style>
div.main div
{
    padding: 15px;
    border: 1px solid black;
}
</style>
</head>
<body>
<div id="maindiv" class="main">
<div ng-controller="maincarcontrol">
<p>{{name}} says that control of this car is {{cartest}}</p>
<div ng-controller="childcarcontrol">
<p>{{name}} says that control of this car is {{cartest}}</p>
<div ng-controller="grandchildcarcontrol">
<p>{{name}} says that control of this car is {{cartest}}</p>
</div>
</div>
</div>
</div>
</body>
</html>

```

Output:

Here you can see that in the first div Anubhav and Awesome are used that were the initial value in maincarcontrol but in the second div Mohit and Awesome are used, Mohit was declared as the initial value but Awesome was not declared, it is still showing it, that's because it is inherited from its parent controller.

In the third div again various values are shown, that's because they were declared in grandchildcarcontrol.

So, if the child control has its own values then they will hide the parent properties and methods.

Using Ng-init and Ng-repeat Directive of AngularJS in ASP.Net Application

Introduction

This explains how to use the ng-init and ng-repeat directives of AngularJS in an ASP.NET Application.

A template is instantiated by ng-repeat for each item present in a collection. A specific scope is provided to each template. In this scope a variable is set to the current collection item.

ng-init is used only with ng-repeat, it is used for aliasing special properties of ng-repeat.

Step 1 - For making today's application you need to add two external files, one is angular.min.js and the second is angular-animate.min.js.

After downloading these files you need to add them to the beginning of the application, in other words in the head tag.

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

Step 2 - Now we will first declare some default values that will be used to show how to use the repeat directive. These values will be provided using the init directive.

```
<div ng-init="cars = [
  {name:'Honda City', buy_in:2003, price:700000},
  {name:'Laura', buy_in:2005, price:1300000},
  {name:'Cruze', buy_in:2010, price:1700000},
  {name:'Audi Q6', buy_in:2013, price:5200000}
]">
```

Here I initialized the value to the Scope using the init directive. Three columns are created in which four different Rows of values are defined.

Step 4 - Now I will show how the ng-repeat directive can be used and how it works.

```
<h2> I have {{cars.length}} cars and they are:</h2>
<ul class="example-animate-container">
  <li class="animate-repeat" ng-repeat="car in cars">
    [{{$index + 1}}] {{car.name}} was bought in {{car.buy_in}} at a price of {{car.price}}.
  </li>
</ul>
</div>
```

Here first I am showing the number of rows that have the data, this can be found using "length", then I took an unordered list in which a List is taken, in this list ng-repeat is used. ng-repeat will help us to show all the data using a list only for one time.

In ng-repeat "car in cars" is declared, cars were declared in the init but car is a variable and you can define it with any name you like.

Then [{{\$index+1}}] is used, this will act as a serial number for each row, it will be automatically incremented on each new row added, after this car.name, car.buy_in and car.price are used that will show the data for a different row.

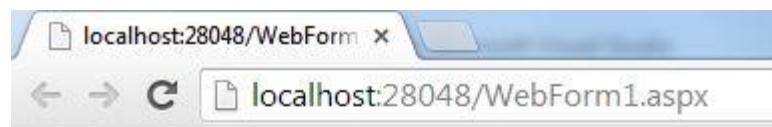
Now our application is ready for execution.

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
  <script src="angular.min.js"></script>
  <script src="angular-animate.min.js"></script>
</head>
<body>
  <form id="form1" runat="server">
    <div ng-init="cars = [
      {name:'Honda City', buy_in:2003, price:700000},
      {name:'Laura', buy_in:2005, price:1300000},
      {name:'Cruze', buy_in:2010, price:1700000},
      {name:'Audi Q6', buy_in:2013, price:5200000}
    ]">
      <h2> I have {{cars.length}} cars and they are:</h2>
      <ul class="example-animate-container">
```

```
<li class="animate-repeat" ng-repeat="car in cars">
  [{{$index + 1}}] {{car.name}} was bought in {{car.buy_in}} at a price of {{car.price}}.
</li>
</ul>
</div>
</form>
</body>
</html>
```

Output: On running the application you will get output like this:



I have 4 cars and they are:

-
- [1] Honda City was bought in 2003 at a price of 700000.
 - [2] Laura was bought in 2005 at a price of 1300000.
 - [3] Cruze was bought in 2010 at a price of 1700000.
 - [4] Audi Q6 was bought in 2013 at a price of 5200000.
-

Filter Data in ASP.Net Application Using AngularJS

Introduction

This explains how to filter data in an ASP.NET Application using AngularJS.

I am working on the previous application so I'll show the data using the repeat and will subscribe to the data using the init directive.

Use the following procedure to create such an application.

Step 1 - For making today's application you need to add two external files, one is angular.min.js and the second is angular-animate.min.js.

After downloading these files you need to add them to the start of the application, in other words in the head tag.

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

Step 2 - Now we will first declare some default values that will be used to show how to use the repeat directive. These values will be provided using the init directive as in the following:

```
<div ng-init="cars = [
  {name:'Honda City', buy_in:2003, price:700000},
  {name:'Laura', buy_in:2005, price:1300000},
  {name:'Cruze', buy_in:2010, price:1700000},
  {name:'Audi Q6', buy_in:2013, price:5200000}
]">
```

Here I have initialized the value to the Scope using the init directive. Three different columns are created in which four different rows of values are defined.

Step 4 - Now I will show how the ng-repeat directive can be used and how it works.

```
<h2> I have {{cars.length}} cars and they are:</h2>
```

```

<label>Type Here to Filter Data: </label><input type="search" ng-
model="q" placeholder="filter cars..." />
<br /><br />
<ul class="example-animate-container">
<li class="animate-repeat" ng-repeat="car in cars | filter:q">
  [{{$index + 1}}] {{car.name}} was bought in {{car.buy_in}} at a price of {{car.price}}.
</li>
</ul>
</div>

```

Here first I am showing the number of rows that have the data, this can be found using ".length", then I took an unordered list in which a list is taken, in this list ng-repeat is used. ng-repeat will help us to show all the data by using a list only for one time.

In the input tag you can see that a ng-model is declared and some text is provided using the placeholder that will be hidden when the user will clicks on the TextBox.

In ng-repeat "car in cars" is declared; cars was declared in the init, in the ng-repeat one more thing is written after the car in cars, in other words "filter:q", this will help to filter the data depending on data provided in the TextBox because "q" was declared in the TextBox.

Then [{{\$index+1}}] is used, this will act as a serial number for each row, it will be auto-incremented on each new row added. After this car.name, car.buy_in and car.price are used that will show the data for a different row.

Now our application is ready to be executed.

The complete code of this application is as follows:

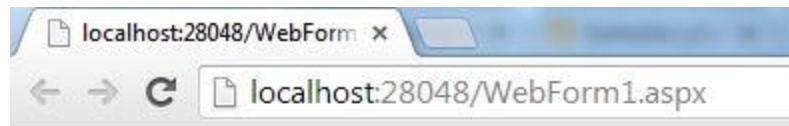
```

<!DOCTYPE html>
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
  <script src="angular.min.js"></script>
  <script src="angular-animate.min.js"></script>
</head>
<body>
  <form id="form1" runat="server">
    <div ng-init="cars = [

```

```
{name:'Honda City', buy_in:2003, price:700000},  
 {name:'Laura', buy_in:2005, price:1300000},  
 {name:'Cruze', buy_in:2010, price:1700000},  
 {name:'Audi Q6', buy_in:2013, price:5200000}  
]>  
<h2> I have {{cars.length}} cars and they are:</h2>  
<label>Type Here to Filter Data: </label><input type="search" ng-  
model="q" placeholder="filter cars..." />  
<br /><br />  
<ul class="example-animate-container">  
<li class="animate-repeat" ng-repeat="car in cars | filter:q">  
 [{{$index + 1}}] {{car.name}} was bought in {{car.buy_in}} at a price of {{car.price}}.  
</li>  
</ul>  
</div>  
</form>  
</body>  
</html>
```

Output: On running the application you will get output like this:

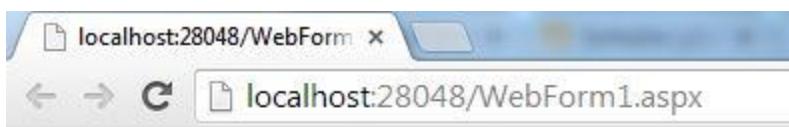


I have 4 cars and they are:

Type Here to Filter Data:

- [1] Honda City was bought in 2003 at a price of 700000.
- [2] Laura was bought in 2005 at a price of 1300000.
- [3] Cruze was bought in 2010 at a price of 1700000.
- [4] Audi Q6 was bought in 2013 at a price of 5200000.

Here you can see that all the data is shown but as I write something in the TextBox, the filter begins to work and filter the data depending on the data entered in the TextBox.



I have 4 cars and they are:

Type Here to Filter Data: ×

- [1] Laura was bought in 2005 at a price of 1300000.
- [2] Audi Q6 was bought in 2013 at a price of 5200000.

Alert Box in AngularJS

Introduction

This topic explains how to show an Alert Box in AngularJS.

There is one basic difference between JavaScript and AngularJS, in other words in JavaScript default names of properties and objects are available for global Window Objects but in AngularJS "\$window" is needed for referring to global Window Objects. For example if we want to show the alert in Angular then we need to pass it as \$window.alert.

This will show you how to display an alert box using AngularJS.

Step 1 - First of all you need to add an external Angular.js file to your application, for this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
<script src="angular.min.js"></script>
<title></title>
</head>
```

Step 2 - Now I will create a function and will show how to create the alert in this function.

Write this function in the head section of your application:

```
<script>
function greet($window, $scope) {
    $scope.hello = function () {
        $window.alert('Hi!! ' + $scope.name);
    }
}
</script>
```

Here I have created a function named "greet", in this function \$scope and \$window are passed as

objects because scope is used to pass a value to the controller and as you know we will display an alert window, that's why \$window is used.

Then an alert is used but has \$window prior to it, in this alert the message "Hi!!" will be displayed along with some value that will be passed through a TextBox. This will be done using the \$scope.name. Now you will be thinking that I have neither declared name anywhere nor provided an initial value to the scope, you can pass the initial value prior to this hello function but that will only help to show some text by default and nothing much more than that.

Step 3 - Until now our work on ViewModel, in other words the Script part is complete and now we need to work on our View, in other words the design part of our application.

Write this code in the body section:

```
<body>
<div ng-controller="greet">
  Name: <input ng-model="name" type="text"/>
  <button ng-click="greet()">Greet</button>
</div>
</body>
```

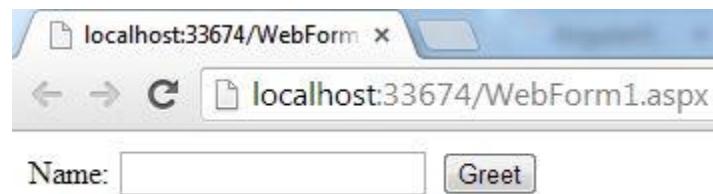
Here first I have applied a controller to a div, then an input tag is used in which "name" is applied through the ng-model, so whatever is entered into the TextBox will be shown in the alert box on the click of a button. The button click is bound to greet using the ng-click().

Now our work on this application is completed and its complete code is as follows:

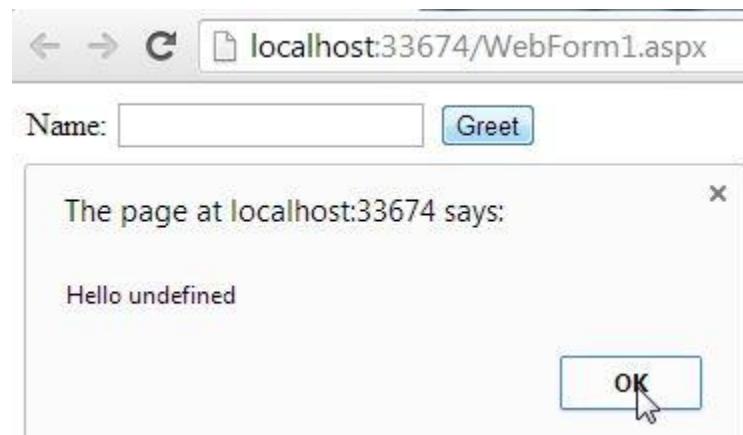
```
<head runat="server">
<script src="angular.min.js"></script>
<title></title>
<script>
  function greet($window, $scope) {
    $scope.hello = function () {
      $window.alert('Hi!! ' + $scope.name);
    }
  }
</script>
```

```
</head>
<body>
<div ng-controller="greet">
  Name: <input ng-model="name" type="text"/>
  <button ng-click="greet()">Greet</button>
</div>
</body></html>
```

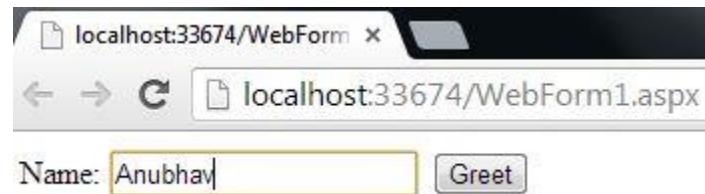
Output: On running the application you will see a TextBox and a Button.



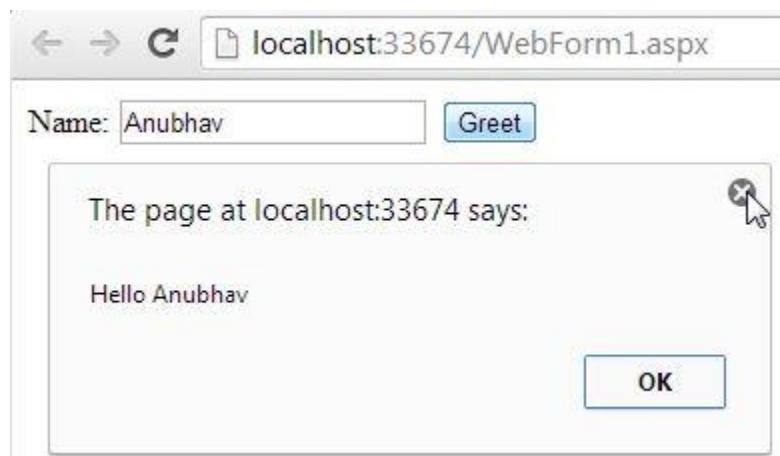
If now I click on the Button then an Alert message will be shown but in the alert box "Hello Undefined" will be shown because nothing is provided through the TextBox and no initial value was passed through the \$scope.



Now I am entering my name in the TextBox:



After entering my name I clicked on the Button to show the alert message and you can see that Hello is printed along with the name entered in the TextBox.



ng-change Directive of AngularJS

Introduction

In this I will tell you about the ng-change directive of AngularJS.

Step 1 - First of all you need to add an external Angular.js file to your application, for this you can go to the AngularJS official site.

After downloading the External File you need to add this file to the Head section of your application.

```
<head runat="server" >
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now I will work on the ViewModel or JavaScript part of this application, here I will create a function and pass some values through the \$scope.

```
<head runat="server" >
    <title></title>
    <script src="angular.min.js"></script>
<script>
    function x($scope) {
        $scope.clickCounter = 0;
        $scope.check = function () {
            $scope.clickCounter++;
        };
    }
</script>
</head>
```

You can see that first I created a function named x(), in this function some initial values are passed, clickCounter is set to 0 at the start.

Then a "check" function is created that will increase the value of clickCounter every time it is triggered.

Our work on ViewModel is complete. Now we just need to work on the View part or design part of this application.

Step 3 - Write this code in the Body tag:

```
<body>
<div ng-controller="x">
<input type="checkbox" ng-model="model" ng-change="check()" id="text1" />
<input type="checkbox" ng-model="model" id="text2" />
<label for="text2">I am Working for Second CheckBox</label><br />
Checkboxes are Checked = {{model}}<br />
<div> counter = {{clickCounter}}</div>
</div>
</body>
```

Here you can see that first I have passed the function in the ng-controller of the Div.

After that I created two Checkboxes, in both the Checkboxes you can see that I passed the ng-model that is necessary if you want to use the ng-change directive in your application, but I had applied ng-change to Checkbox1 so the click counter will be incremented only on the click of the first checkbox.

After that, two labels are used, with the first Label "model" is bound and since it was bound to checkboxes it will either show a True value or a False value.

After this "clickCounter" is bound in a Div that will show the increment of the click counter. Now our application is ready to be executed.

The complete code of this application is as follows:

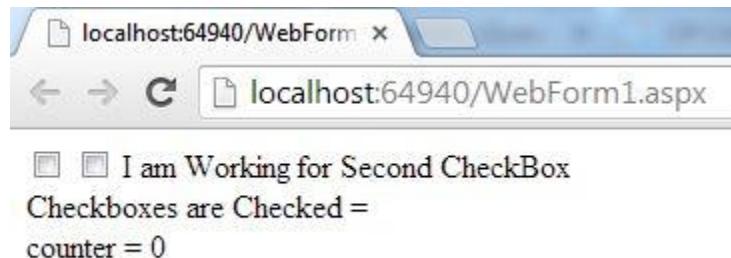
```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server" >
<title></title>
<script src="angular.min.js"></script>
<script>
function x($scope) {
```

```

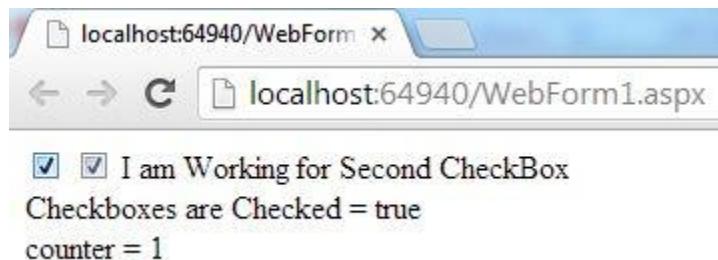
$scope.clickCounter = 0;
$scope.check = function () {
    $scope.clickCounter++; }
}
</script>
</head>
<body>
<div ng-controller="x">
<input type="checkbox" ng-model="model" ng-change="check()" id="text1" />
<input type="checkbox" ng-model="model" id="text2" />
<label for="text2">I am Working for Second CheckBox</label><br />
Checkboxes are Checked = {{model}}<br />
<div> counter = {{clickCounter}}</div>
</div>
</body>
</html>

```

Output: On running the application you will get output like this:

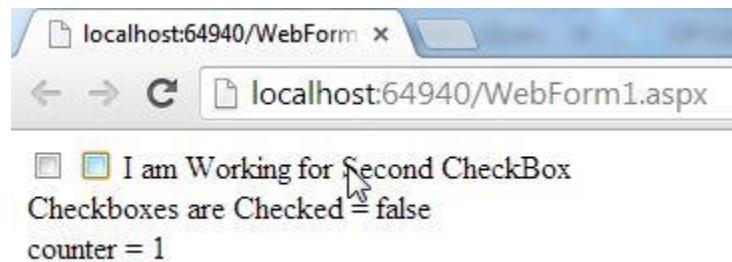


As you can see, at this moment everything is in its initial state, but as I click the first checkbox changes can be seen everywhere.

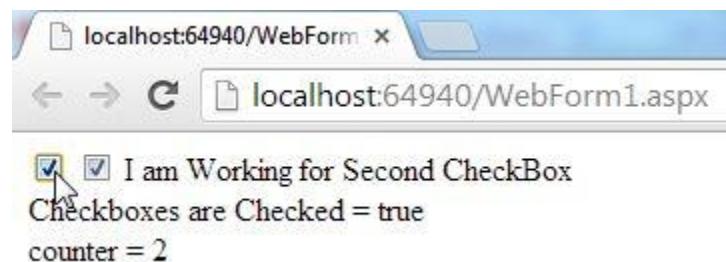


You can see that the click counter is incremented by one, the second checkbox is automatically checked because both have the same model name, true is shown because the checkboxes are checked.

Now if I click on the Label present beside the second checkbox then you will see that it is also making the checkboxes checked and unchecked.



This label is working for checkbox 2 because I have defined in my code as for="text2". You can see that "False" is also shown because the checkboxes are unchecked but the counter is not incremented because ng-change was only bound to the first checkbox.



Now again as I click on first TextBox you can see that the counter is again incremented by one.

Ng-click and Ng-blur Directives in AngularJS

Introduction

In this I will tell you about the ng-click and ng-blur directives in AngularJS.

Step 1 - First of all you need to add an external Angular.js file to your application, for this you can go to the AngularJS official site. After downloading the external file you now need to add this file to the Head section of your application as in the following:

```
<head runat="server">
<script src="angular.min.js"></script>
<title></title>
</head>
```

Step 2 - First I will work on the click functionality of a button. Write this code in the Body section of your application:

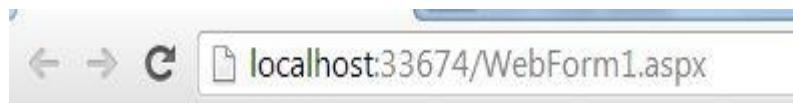
```
<div>
<h2>You have clicked the Button: {{click}} Times</h2>
<button ng-click="click=click+1" ng-init="click=0" >Increment</button>
</div>
```

I will create a click counter that will work on the click of a button; for that I took a button and a label in a Div.

In the button you can see that I had first provided the initial value in the init that is set to "0", in the button ng-click is also used that will increment the value of click by 1 on each new click.

The value of the click counter is provided in the H2 tag, in the H2 tag you can see that I bound it with the click using {{ }}.

Now if we run our application than first 0 will be shown by the H2 tag:



You have clicked the Button: 0 Times

Now if I click on the Button than you will see that the value is regularly incremented by 1.



You have clicked the Button: 4 Times

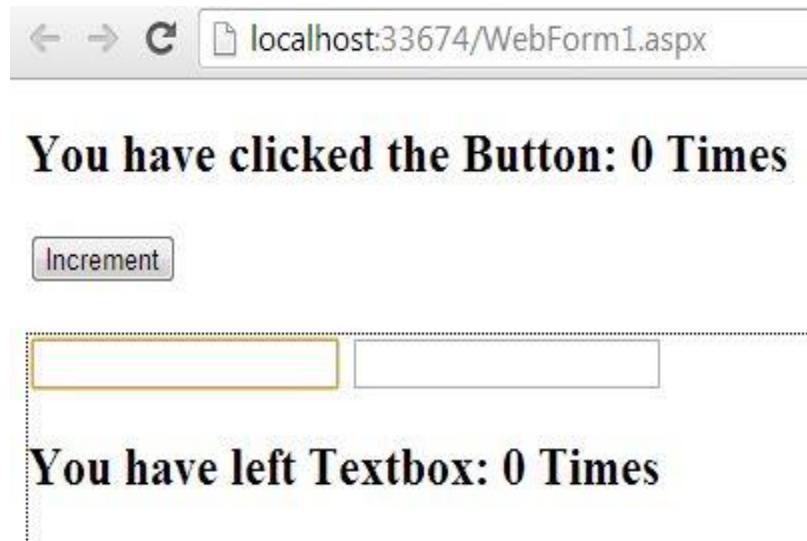
In the next steps I will show you how to use the ng-blur directive of AngularJS.

Step 3 - I will again create a click counter but this time it will work on ng-blur() and not on ng-click(). Since I am showing the Blur() the best way of showing is using a TextBox. I took two Textboxes but ng-blur will be applied to the first TextBox only.

```
<div style="border:1px dotted">
<input ng-blur="leave=leave+1" ng-init="leave=0" />
<input id="text2" />
<h2>You have left Textbox: {{leave}} Times</h2>
</div>
```

Here again I had passed some initial value through the init, this value will be incremented by one on blur.

Now if we run our application then first 0 will be shown by the H2 tag:



You have clicked the Button: 0 Times

Increment

You have left Textbox: 0 Times

As the first TextBox loses its focus you can see that the counter starts incrementing by 1.



You have clicked the Button: 0 Times

Increment

You have left Textbox: 1 Times

The complete code is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <script src="angular.min.js"></script>
    <title></title>
</head>
<body>
    <div>
        <h2>You have clicked the Button: {{click}} Times</h2>
        <button ng-click="click=click+1" ng-init="click=0" >Increment</button>
    </div>
    <br />
    <div style="border:1px dotted">
        <input ng-blur="leave=leave+1" ng-init="leave=0" />
        <input id="text2" />
        <h2>You have left Textbox: {{leave}} Times</h2>
    </div>
</body>
</body>
</html>
```

Required Field and Email Validation Using AngularJS

Introduction

This explains Required Field Validation and Email Validation using AngularJS.

AngularJS provides various kinds of validations that can be used in applications, in this article I will show only two types of validations provided by AngularJS, in other words Required Field and Email Validation.

Let's see the procedure to apply validations using AngularJS.

Step 1 - First of all you need to download the AngularJS external file; that can be done from the jQuery official website. After downloading the file you need to add it in the Head Section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now I will first show how to implement Required Field Validation using AngularJS.

First we will create a function in which one initial value will be passed that will be shown by default whenever the user visits the page.

This function can be created in following manner:

```
<script>
    function x($scope) {
        $scope.initialname = "Anu";
    }
</script>
```

Here I have passed an initial value to the scope as "initialname"

Step 3 - Now we will work on the design part where validation will be applied.

Write this code in the body section of your web page:

```
<body>
<div ng-app>
<form name="form1" ng-controller="x">
<label>Your Name:</label>
<input name="name" ng-model="initialname" required>
<span style="color:red" ng-show="form1.name.$error.required">
Please Provide Your name</span>
<br />
<br />
<input type="submit" value="Submit"/>
</form>
</div>
</body>
```

Here first you need to add ng-app to the Div tag otherwise Validations will not work properly. Then a name is provided to the form and the function name is passed in the ng-controller of the form.

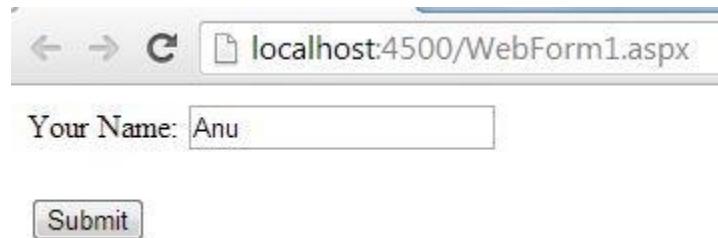
Then a TextBox is created to which again a name is provided, in this the TextBox initial value is also bound using the ng-model.

Then a span is created in which ng-show is used, in this ng-show an error message is provided that will be activated when the user does not provide a value in the TextBox. This error message is provided in the following manner.

The first name of the form is provided, then this name of the TextBox is provided, then this \$error is used that shows that it will be activated on getting an error and the final word is written as required that shows that it will check whether some value is provided or not.

At the end I took a button. Now our application is created and is ready to be executed.

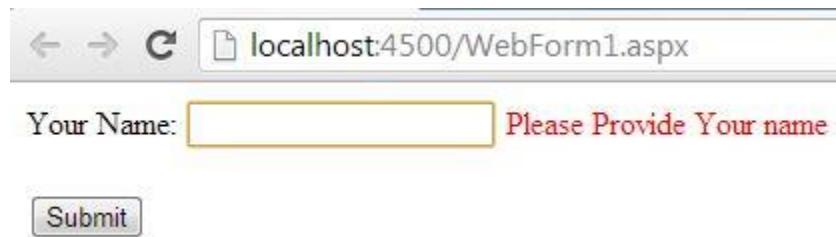
Output: On running the application you will see that the initial value is shown in the TextBox.



localhost:4500/WebForm1.aspx

Your Name:

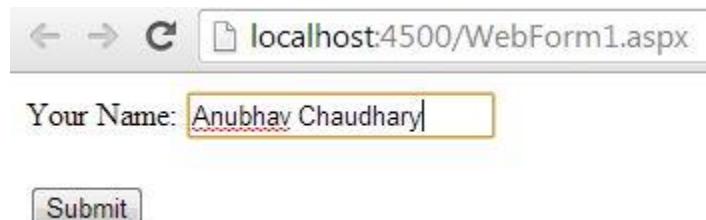
But if I remove this initial value than an error message will be displayed just beside this TextBox.



localhost:4500/WebForm1.aspx

Your Name: Please Provide Your name

Now I am writing my complete name in this TextBox and you will see that the error message is gone.



localhost:4500/WebForm1.aspx

Your Name:

Until now you have seen the Required Field Validation, now I will show you how to apply Email Validation using AngularJS.

Step 4 - I am adding an initial value for Email; as well, it will be added to the function that was present in the Head Section. So, now your function will look like this:

```
<script>
function x($scope) {
    $scope.initialname = "Anu";
```

```

$scope.initialmail = "test@test.com";
}
</script>

```

Here "initialmail" contains the initial value that will be shown in the TextBox by default.

Step 5 - After this I had added one more TextBox and Span in the body section that will be working for the Email.

So, now body Section is modified to be:

```

<body>
  <div ng-app>
    <form name="form1" ng-controller="x">
      <label>Your Name:</label>
      <input name="name" ng-model="initialname" required>
      <span style="color:red" ng-show="form1.name.$error.required">
        Please Provide Your name</span>
      <br />
      <br />
      <label>Your Email: </label>
      <input type="email" name="email" ng-model="initialmail">
      <span style="color:red" ng-show="form1.email.$error.email">
        Please Provide valid EmailID</span>
      <br />
      <br />
      <input type="submit" value="Submit"/>
    </form>
  </div>
</body>

```

You can see that everything is similar except that in the Span the email is written instead of required.

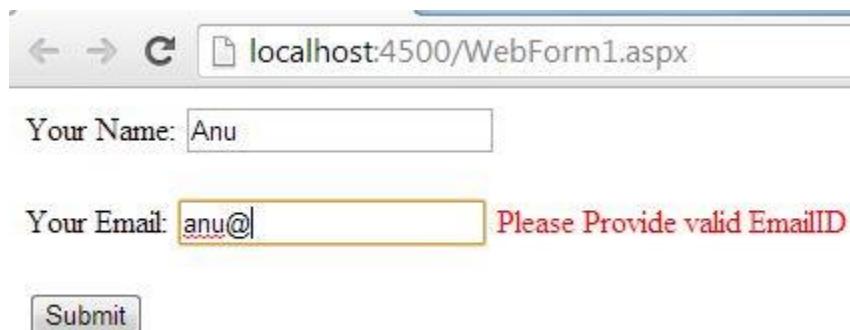
Our application is ready to be executed.

Output: On running the application you will see that the initial value is shown in both of the Textboxes.



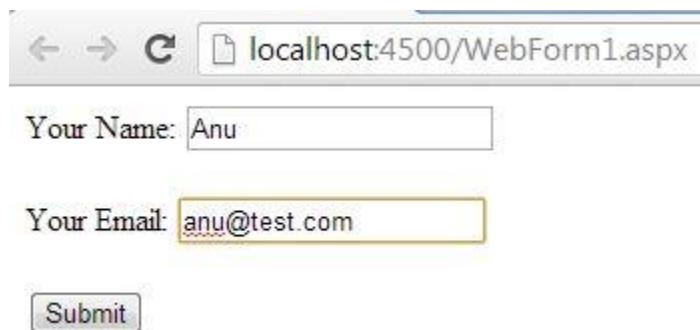
A screenshot of a web browser window. The address bar shows "localhost:4500/WebForm1.aspx". The page contains two text input fields: "Your Name: Anu" and "Your Email: test@test.com", both in light blue. Below them is a "Submit" button.

Now If I provide an incorrect EmailId then an Error message will be shown just beside the second TextBox.



A screenshot of a web browser window. The address bar shows "localhost:4500/WebForm1.aspx". The page contains two text input fields: "Your Name: Anu" and "Your Email: anu@". To the right of the email field, the text "Please Provide valid EmailID" is displayed in red. Below the fields is a "Submit" button.

Now I am providing the Email Id in correct format and you can see that the error message is gone.



A screenshot of a web browser window. The address bar shows "localhost:4500/WebForm1.aspx". The page contains two text input fields: "Your Name: Anu" and "Your Email: anu@test.com", both in light blue. Below the fields is a "Submit" button.

Minimun Length and Maximum Length Validation Using AngularJS

Introduction

In this I will tell you about Min Length and Max Length Validation using AngularJS.

AngularJS provides many kinds of validations that can be used in our applications, in this article I will show only two types of validations provided by AngularJS, in other words Required Field Validation and Email Validation.

Let's see the procedure to apply validations using AngularJS.

Step 1 - First of all you need to download the AngularJS external file that can be done from the jQuery official website.

After downloading the file you need to add it in the Head Section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 – Now I will first show how to implement Min Length Validation using AngularJS.

First we will create a function in which one initial value will be passed that will be shown by default whenever the user visits the page.

This function can be created in the following manner:

```
<script>
    function x($scope) {
        $scope.initialname = "Anu";
    }
</script>
```

Here I had passed an initial value to the scope as “initialname”.

Step 3 - Now we will work on the design part where the validation is to be applied.

Write this code in the body section of your web page:

```
<body>
<div ng-app>
<form name="form1" ng-controller="x">
  <label>Your Name:</label>
  <input name="name" ng-model="initialname" ng-minlength="3">
  <span style="color:red" ng-show="form1.name.$error.minlength">
    Your Name Should Contain Atleast 3 Characters</span>
  <br />
  <br />
  <input type="submit" value="Submit"/>
</form>
</div>
</body>
```

Here first you need to add ng-app to the Div tag otherwise the validations will not work properly. Then a name is provided to the form and the function name is passed in the ng-controller of the form.

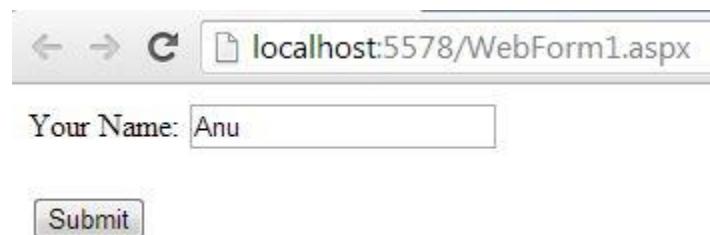
Then a TextBox is created to which again a name is provided, in this TextBox an initial value is also bound using ng-model, then you can see that I have provided the minimum length as ng-minlength="3" this means that if the user enters a value consisting of less than 3 characters then an error message will be shown.

Then a span is created in which ng-show is used, in this ng-show an error message is provided that will be activated when the user does not provide a value in the TextBox. This error message is provided in the following manner:

First the name of the form is provided, then the name of the TextBox is provided, then \$error is used that shows that it will be activated on getting an error and the final word is written as minlength that shows that it will check whether the user has provided a name consisting of less than 3 letters, if the name is contains less than 3 letters then the error will be shown.

At the end I took a button. Now our application is created and is ready to be executed.

Output: On running the application you will see that the initial value is shown in the TextBox.



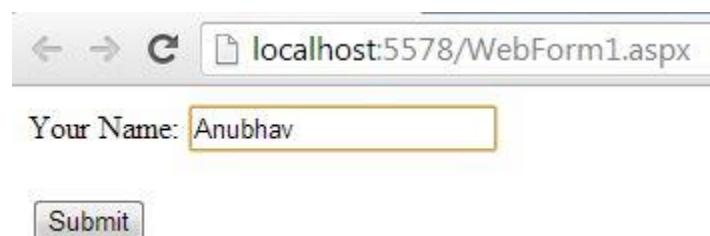
A screenshot of a web browser window. The address bar shows "localhost:5578/WebForm1.aspx". Below the address bar is a form with a label "Your Name:" followed by a text input field containing the text "Anu". Below the input field is a "Submit" button.

But as I make changes in this initial value and remove a letter you can see that an error message is shown that your name should contain at least 3 letters.



A screenshot of a web browser window. The address bar shows "localhost:5578/WebForm1.aspx". Below the address bar is a form with a label "Your Name:" followed by a text input field containing the text "An". To the right of the input field, the error message "Your Name Should Contain Atleast 3 Characters" is displayed in red. Below the input field is a "Submit" button.

Now If I provide my complete name than the error is gone.



A screenshot of a web browser window. The address bar shows "localhost:5578/WebForm1.aspx". Below the address bar is a form with a label "Your Name:" followed by a text input field containing the text "Anubhav". There is no error message displayed. Below the input field is a "Submit" button.

Until now you saw the MinLength Validation. I will now show you how to apply MaxLength Validation using AngularJS.

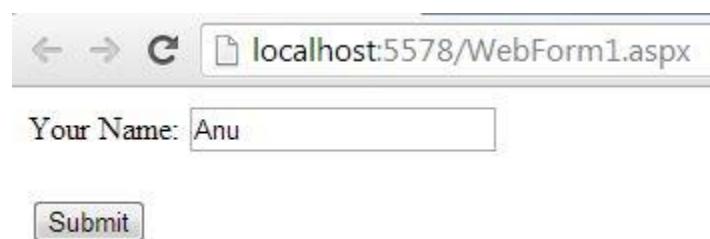
Step 4 - For this you need to go to the body section and make changes in the TextBox that was already created, you just need to Add "ng-maxlength="20"" and maxlength validation will be applied, so now your code will look like:

```
<body>
  <div ng-app>
    <form name="form1" ng-controller="x">
      <label>Your Name:</label>
      <input name="name" ng-model="initialname" ng-minlength="3" ng-maxlength="20">
      <span style="color:red" ng-show="form1.name.$error.minlength">
        Your Name Should Contain Atleast 3 Characters</span>
      <span style="color:red" ng-show="form1.name.$errormaxlength">
        Sorry You are Exceeding the Limit</span>
      <br />
      <br />
      <input type="submit" value="Submit"/>
    </form>
  </div>
</body>
```

In the TextBox you can see that I had added ng-maxlength="20", but I had added one more span in which an error message for the max length will be shown.

Now our application is ready to be executed.

Output: On running the application you will see that the initial value is shown in the TextBox.



But as I make changes in this initial value and provide a name consisting of more letters then you will see the error message shown that your have exceeded the limit.



Custom Validation Using AngularJS

Introduction

In this I will tell you about Custom Validation using AngularJS.

AngularJS provides various validations that can be used in our applications, in this I will show only Custom Validations provided by AngularJS. Let's see the procedure required to apply validations using AngularJS.

Step 1 - First of all you need to download the AngularJS external file. That can be done in jQuery official website.

After downloading the file you need to add it to the Head Section of your application.

```
<head id="Head1" runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - First we will create a function in which one initial value will be passed to be shown by default whenever the user visits the page. Also a pattern will be provided to be matched when inserting the value into the TextBox.

This function can be created in the following manner:

```
<head id="Head1" runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.initialname = "123";
            $scope.pattern = /^d*/;
        }
    </script>
</head>
```

Here I have passed an initial value to the scope as "initialname".

The pattern is passed using the "pattern" in the scope, here the pattern will ensure that only Integers are entered in the TextBox.

Step 3 - Now we will work on the design part where the validation will be applied.

Write this code in the body section of your web page:

```
<body>
<div ng-app>
<form name="form1" ng-controller="x">
  <label>Your Name:</label>
  <input name="name" ng-model="initialname" ng-pattern="pattern">
  <span style="color:red" ng-show="form1.name.$error.pattern">
    Please Insert Numbers Which Should be Positive</span>
  <br />
  <br />
  <input type="submit" value="Submit"/>
</form>
</div>
</body>
```

Here first you need to add "ng-app" to the Div tag otherwise Validations will not work properly. Then a name is provided to the form and the function name is passed in the ng-controller of the form.

Then a TextBox is created to which again a name is provided. In this TextBox an initial value is bound using the ng-model and the pattern is bound using the ng-pattern.

Then a span is created in which ng-show is used, in this ng-show an error message is provided which will be activated when the user does not provide a valid number in the TextBox. This error message is provided in the following manner.

First the name of the form is provided, then the name of the TextBox is provided, then this \$error is used that shows that it will be activated on getting an error and the final word is written as a pattern showing that it will check whether the user has provided a valid number or not, if the number is negative or not a number than an error will be shown.

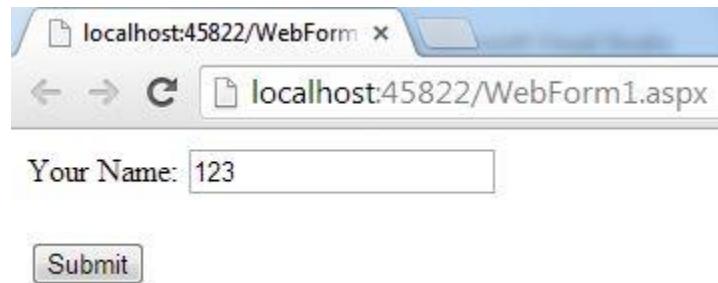
Finally I used a button.

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.initialname = "123";
            $scope.pattern = /^\\d*/;
        }
    </script>
</head>
<body>
    <div ng-app>
        <form name="form1" ng-controller="x">
            <label>Your Name:</label>
            <input name="name" ng-model="initialname" ng-pattern="pattern">
            <span style="color:red" ng-show="form1.name.$error.pattern">
                Please Insert Numbers Which Should be Positive</span>
            <br />
            <br />
            <input type="submit" value="Submit"/>
        </form>
    </div>
</body>
</html>
```

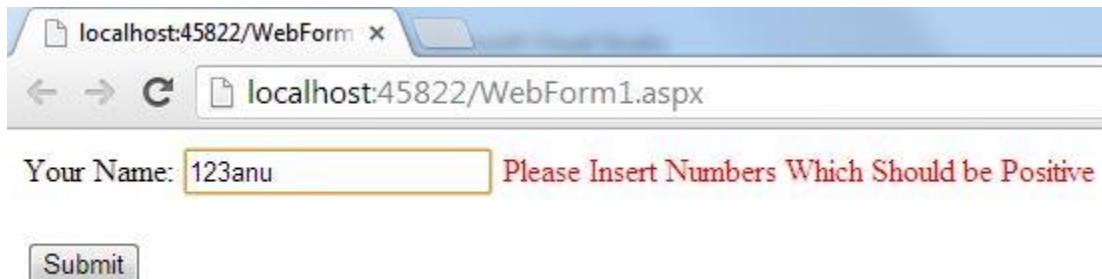
Now our application is created and is ready to be executed.

Output: On running the application you will see that the initial value is shown in the TextBox.



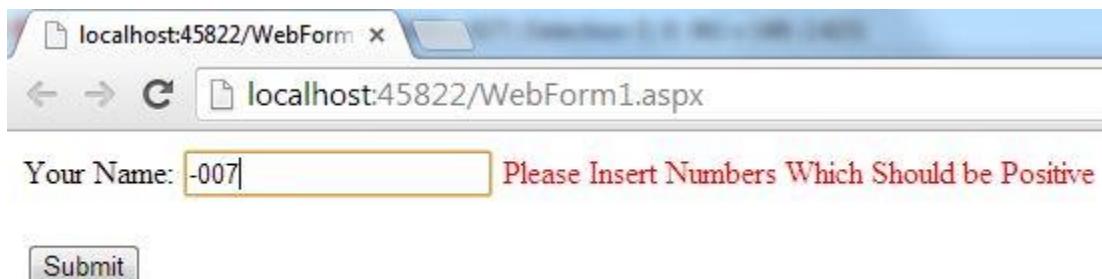
A screenshot of a web browser window titled "localhost:45822/WebForm". The address bar shows "localhost:45822/WebForm1.aspx". The page contains a single input field with the placeholder "Your Name: 123" and a "Submit" button below it.

Now if I enter some letters into this TextBox then you will see that the error message will be shown immediately.



A screenshot of a web browser window titled "localhost:45822/WebForm". The address bar shows "localhost:45822/WebForm1.aspx". The page contains a TextBox with the value "123anu" and an error message "Please Insert Numbers Which Should be Positive" displayed next to it, and a "Submit" button below.

Also if I provide a negative integer than also the error message will be shown.



A screenshot of a web browser window titled "localhost:45822/WebForm". The address bar shows "localhost:45822/WebForm1.aspx". The page contains a TextBox with the value "-007" and an error message "Please Insert Numbers Which Should be Positive" displayed next to it, and a "Submit" button below.

ng-BindHtml Directive of AngularJS

Introduction

In this I will tell you about the ng-BindHtml Directive of AngularJS.

Using ng-BindHtml we can declare a HTML element in the Controller but they work similarly to a normal HTML element.

Step 1 - First of all you need to add two external Angular.js files to your application, in other words:

1. angular.min.js
2. angular-sanitize.min.js

For this you can go to the AngularJS official site. After downloading the External Files now you need to add these files to the Head section of your application.

```
<head runat="server">
<title></title>
<script src="angular.min.js"></script>
<script src="angular-sanitize.min.js"></script>
</head>
```

Step 2 - Now I will work on the ViewModel or JavaScript part of this application, here I am using the second method for initializing AngularJS (I already told you three ways of initializing AngularJS).

Write this code in the Head Section:

```
<head runat="server">
<title></title>
<script src="angular.min.js"></script>
<script src="angular-sanitize.min.js"></script>
<script>
angular.module('htmlDoc', ['ngSanitize']).controller('x', function ($scope) {
    $scope.html = 'You can see that I am <b>Bold </b> and <i>Italic </i> also';
```

```
});
</script>
</head>
```

Here I declared "htmldoc" using the angular.module, this "htmldoc" is the value that will be provided in the ng-app. Then "ngSanitize" is used that is necessary if we are using ng-BindHtml.

Then I have provided the name of the controller as "x", then you can see that I have provided an initial value in which the bold and italic tags are used, in the output you will see that they are working as normal HTML elements work.

Our work on ViewModel is complete and now we just need to work on the View part or design part of this application.

Step 3 - Write this code in the Body tag:

```
<body>
  <form id="form1" runat="server">
    <div ng-controller="x">
      <p ng-bind-html="html"></p>
    </div>
  </form>
</body>
```

Here I had first provided the controller name in the div, then I had bound the HTML document that was created in the head tag with the <p> tag using the ng-bind-html.

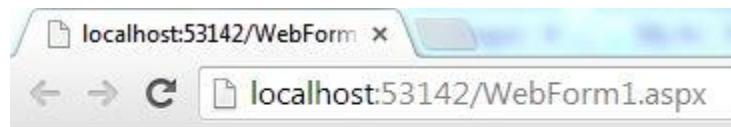
Now our application is ready to be executed.

The complete code of this application is as follows:

```
<html ng-app="htmldoc" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
  <script src="angular.min.js"></script>
  <script src="angular-sanitize.min.js"></script>
  <script>
    angular.module('htmldoc', ['ngSanitize']).controller('x', function ($scope) {
      $scope.html = 'You can see that I am <b>Bold </b> and <i>Italic </i> also';
    });
  </script>
```

```
</script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <p ng-bind-html="html"></p>
        </div>
    </form>
</body>
</html>
```

Output: On running the application you will get an output like this one:

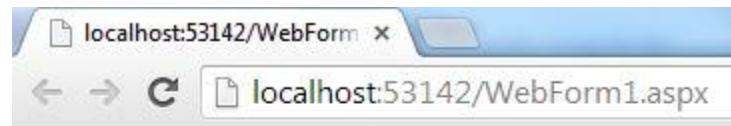


You can see that I am **Bold** and *Italic* also

Now I am making a slight change in the Script, I am adding a `<H2>` tag.

```
<script>
    angular.module('htmlDoc', ['ngSanitize']).controller('x', function ($scope) {
        $scope.html = 'You can see that I am <h2>Heading </h2> and <i>Italic </i> also';
    });
</script>
```

You will see that the output is showing the text as expected.



You can see that I am

Heading

and *Italic* also

Apply Validation For Checking Valid URL Using AngularJS

Introduction

In this I will explain how to apply validation for checking for a valid URL using AngularJS.

Your project might include something where the user needs to provide his website a name or the name of the website that he wants to access, in these cases you require a valid URL of the website from the user, for this to happen you must apply some validation on your form that can stop the user from providing an invalid format of URL, this article will help you apply this validation using AngularJS.

Now I will show you an example of how you can apply this validation.

Step 1 - First of all you need to download an AngularJS external file. That can be done either using the jQuery official website or you can also download the source code that I provided at the top of this article.

After downloading the file you need to add it in the Head Section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - First we will create a function in which one initial Web Site URL will be passed that will be shown by default whenever the user visits the page.

This function can be created in the following manner:

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.initialwebsite = "http://Gmail.com"; } </script></head>
```

Here x is the Controller and \$scope provides the connection between the function and the view of the application.

In this function I had passed an initial value to the scope as "initialwebsite".

Step 3 - Now we will work on the design part where the validation will be applied.

Write this code in the body section of your web page:

```
<body>
<div ng-app>
<form name="form1" ng-controller="x">
<label>Enter Site Name:</label>
<input type="url" ng-model="initialwebsite" name="name" required />
<span style="color:red" ng-show="form1.name.$error.required">
You Can't Leave This Field Empty</span>
<span style="color:red" ng-show="form1.name.$error.url">
Sorry Not a Valid URL, Don't Forget to Use http://</span>
<br />
<br />
<input type="submit" value="Submit"/>
</form>
</div>
</body>
```

Here I first took a TextBox whose type is set to "url", the initial value is applied to this TextBox using ng-model. One name is also provided to this TextBox that will be used while applying the Validation Error Message.

Then I took two Span, in the first Span is an error message for the required field validation that is to be applied so that the user can't leave the TextBox empty. In the second Span is an error message for the correct format of the URL must be applied. It's applied in the following format:

First the name of the Form is provided, after the name of the TextBox is provided, after this \$error is used that shows that it will be activated on getting an error and the final word is written as an URL that shows that it will check whether the user has provided a valid URL or not, if the URL is not valid or the User has forgotten to use http:// than an error will be shown.

At the end I used a button. The complete code of this application is as follows:

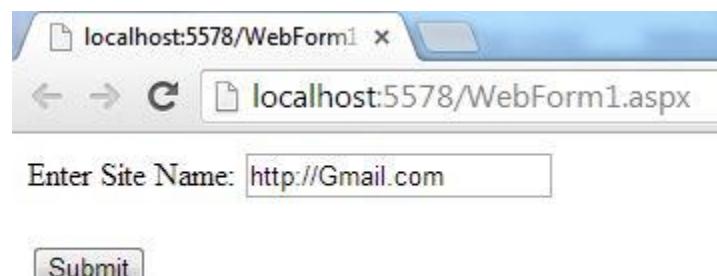
```

<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.initialwebsite = "http://Gmail.com"; }
    </script>
</head>
<body>
    <div ng-app>
        <form name="form1" ng-controller="x">
            <label>Enter Site Name:</label>
            <input type="url" ng-model="initialwebsite" name="name" required />
            <span style="color:red" ng-show="form1.name.$error.required">
                You Can't Leave This Field Empty</span>
            <span style="color:red" ng-show="form1.name.$error.url">
                Sorry Not a Valid URL, Don't Forget to Use http://</span>
            <br />
            <br />
            <input type="submit" value="Submit"/>
        </form>
    </div>
</body>
</html>

```

Now our application is created and is ready for execution.

Output: On running the application you will see that the initial value is shown in the TextBox.



Now if I try to leave the TextBox empty then the required field validator will be activated and will show the error message.



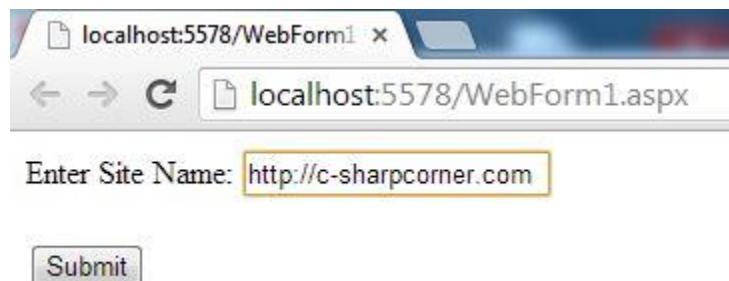
A screenshot of a Microsoft Internet Explorer browser window. The address bar shows 'localhost:5578/WebForm1.aspx'. The page content includes a label 'Enter Site Name:' followed by a yellow-bordered text input field. To the right of the input field is the error message 'You Can't Leave This Field Empty' in red. Below the input field is a 'Submit' button.

Also if I try to provide the URL without using http:// or try to provide the URL in any other inappropriate format then an error message will also be shown to enter the URL in the correct format.



A screenshot of a Microsoft Internet Explorer browser window. The address bar shows 'localhost:5578/WebForm1.aspx'. The page content includes a label 'Enter Site Name:' followed by a yellow-bordered text input field containing 'c-sharpcorner.com'. To the right of the input field is the error message 'Sorry Not a Valid URL, Don't Forget to Use http://' in red. Below the input field is a 'Submit' button.

Now I am providing the URL in the correct format and you can see that all the validation error messages are gone.



A screenshot of a Microsoft Internet Explorer browser window. The address bar shows 'localhost:5578/WebForm1.aspx'. The page content includes a label 'Enter Site Name:' followed by a yellow-bordered text input field containing 'http://c-sharpcorner.com'. Below the input field is a 'Submit' button.

Bind Multiple Values Using AngularJS

Introduction

This explains the ng-bind-template directive of AngularJS.

There is a problem with ng-bind; you can only bind a single {{}} expression with it, now the question is what to do when we want to bind multiple expressions with it?

The solution is ng-bind-template, it can bind more than one {{}} expression, so it can show more than a single value that was declared in the Script function. Now I am showing you an example of how to use this directive in your application.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official Site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now I will work on the ViewModel or JavaScript part of this application.

Write this code in the Head Section of your application:

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.firstName = 'Anubhav';
            $scope.lastName = 'Chaudhary'; }
    </script>
</head>
```

Here I have declared two initial values; the first is the firstName and the second is the secondName. These values will be shown by default whenever the user runs the application.

Our work on the ViewModel is complete, we now just need to work on the View part or design part of this application.

Step 3 - Write this code in the Body tag:

```
<body>
  <form id="form1" runat="server">
    <div ng-controller="x">
      <label>First Name: </label><input type="text" ng-model="firstName" />
      <br />
      <label>Last Name: </label><input type="text" ng-model="lastName" />
      <br />
      <br />
      <label ng-bind-template="Hello! {{firstName}} {{lastName}}"></label>
    </div>
  </form>
</body>
```

Here the Controller Name is the Function name that was declared in the Step 2, this controller name is provided in the Div.

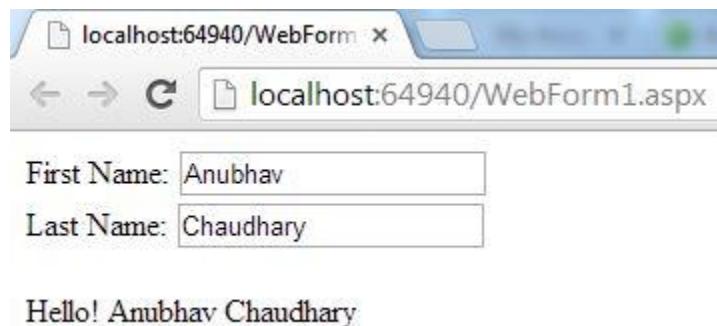
Then both of the initial values are bound to two different Textboxes using ng-model. Towards the end you can see that both the initial values are bound together to the same Label using the ng-bind-template.

Now our application is ready for execution. The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
  <script src="angular.min.js"></script>
  <script>
    function x($scope) {
      $scope.firstName = 'Anubhav';
      $scope.lastName = 'Chaudhary';
    }
  </script>
```

```
</script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <label>First Name: </label><input type="text" ng-model="firstName" />
            <br />
            <label>Last Name: </label><input type="text" ng-model="lastName" />
            <br />
            <br />
            <label ng-bind-template="Hello! {{firstName}} {{lastName}}"></label>
        </div>
    </form>
</body>
</html>
```

Output: On running the application you will get an output like this:



Here you can see that the first and second TextBoxes are showing their initial values that were declared in the Script Tag and both of the values are shown in the Label as well.

Now when I make changes in the TextBox then the result can instantly be seen in the Label present at the end as well.

localhost:64940/WebForm x

localhost:64940/WebForm1.aspx

First Name:

Last Name:

Hello! Mohit Chaudhary

Change Style Dynamically in Various Ways Using AngularJS

Introduction

This explains how to change the style dynamically in various ways using AngularJS.

AngularJS provides three different ways to change the style dynamically. People always search for methods to change the style dynamically, AngularJS can solve their problems in any of three ways.

Step 1 - First of all you need to add an external Angular.js file to your application, for this you can go to the AngularJS official site.

After downloading the external file you need to add this file to the Head section of your application as in the following:

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now I will create some simple CSS classes to be applied dynamically, or you can say at runtime.

```
<style>
    .changeColor {
        color:blue;
    }
    .changeSize {
        font-size:30px;
    }
    .italic {
        font-style:italic;
    }
</style>
```

Here I have created three classes, namely changeColor, changeSize, and italic. The first CSS Class is used for changing the color of the text to Blue, the second is for changing the text size and the last one is for making the Text Italic.

Now we have created the classes to be applied, now we need to work on the ViewModel or design part of our application where the simple coding needs to be done to apply the change in CSS.

First Method-

In the first method objects of the classes are created that are called using the ng-model, let's see this by implementing it in our application as in the following:

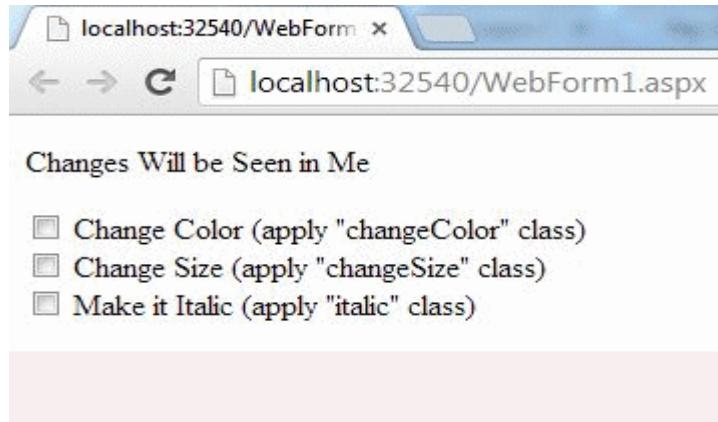
```
<p ng-
class="{changeColor: Color, changeSize: Size, italic: Italic}">Changes Will be Seen in Me</p>
<input type="checkbox" ng-model="Color"> Change Color (apply "changeColor" class)<br>
<input type="checkbox" ng-model="Size"> Change Size (apply "changeSize" class)<br>
<input type="checkbox" ng-model="Italic"> Make it Italic (apply "italic" class)
```

Here I have applied the CSS to a <p> Tag, classes are applied using the ng-class directive, here in the ng-class you can see that I have created an object of each class such as for changeColor, "Color" is created.

After this I had applied the binding using the ng-model, I have created three checkboxes and in each checkbox ng-model is applied, each ng-model is providing the binding using the objects created in the ng-class.

So, If we check the first checkbox then the color of the text will change to Blue, similarly if we check the second and third checkboxes then the text size and its style will change.

Output: As I run the application and check the checkboxes then this type of output will be seen:



Second Method-

Now, I will show you the Second Method of applying the CSS dynamically.

In this method Classes Names will be called to Apply the CSS.

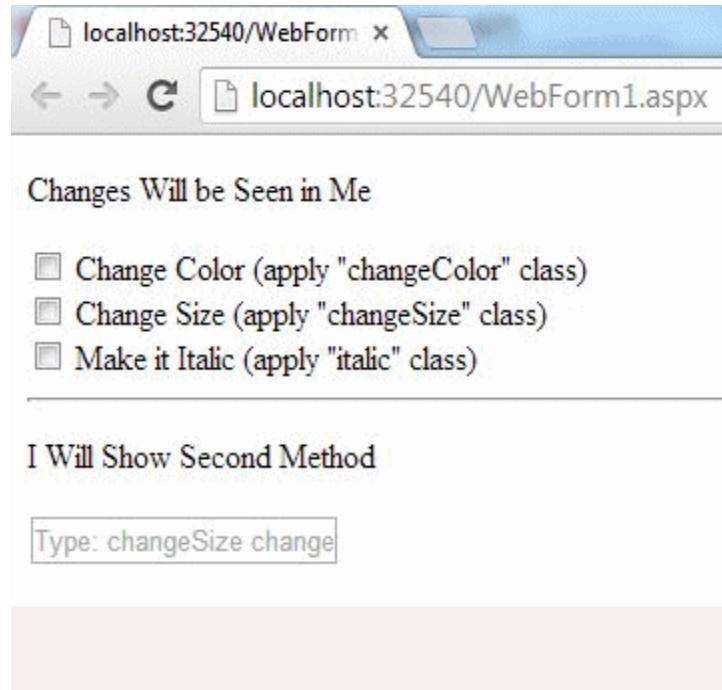
```
<p ng-class="style">I Will Show Second Method</p>
<input type="text" ng-model="style" placeholder="Type: changeSize changeColor italic">
```

Here you can see that in the ng-class I have passed the complete style, since I have passed the style in ng-class it will call all the classes that are created in that style.

Then I have created a TextBox to which the style is bound using the ng-model, whatever name is provided in the TextBox, the same style will be applied to the text provided through the <p> tag. If the the name provided in the TextBox doesn't match the CSS Class than nothing will happen to the text and nothing will be applied.

Let's see the output.

Output:



The screenshot shows a web browser window with the URL `localhost:32540/WebForm1.aspx`. The page content includes:

- A heading "Changes Will be Seen in Me" followed by three checkboxes:
 - Change Color (apply "changeColor" class)
 - Change Size (apply "changeSize" class)
 - Make it Italic (apply "italic" class)
- A horizontal line separator.
- A heading "I Will Show Second Method" followed by a text input field containing the placeholder "Type: changeSize change".
- A large pink rectangular background element below the input field.

Third Method-

Now, I will show you the Third Method for applying the CSS dynamically.

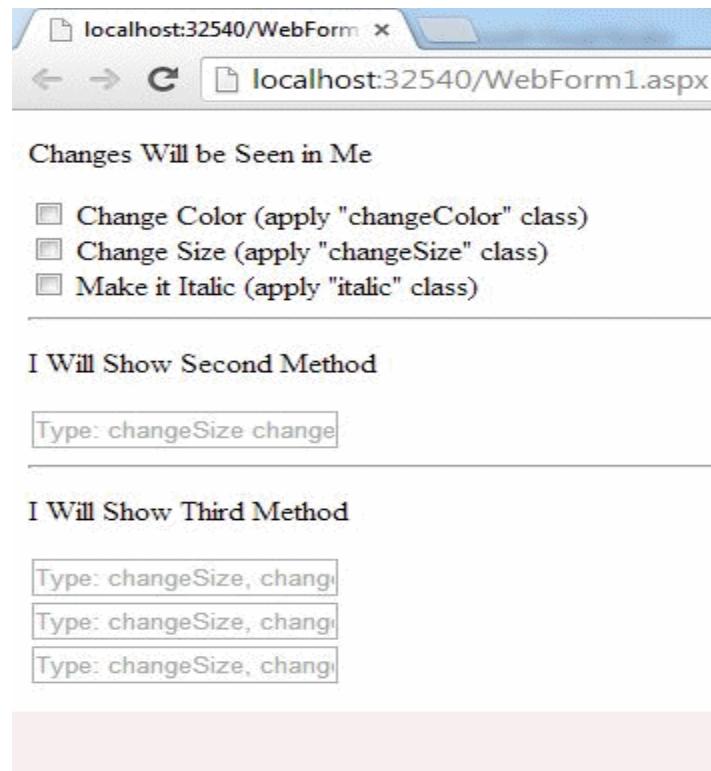
In this method CSS classes will be called in the order they were created.

```
<p ng-class="[style1, style2, style3]">Using Array Syntax</p>
<input ng-model="style1" placeholder="Type: changeSize, changeColor or italic"><br>
<input ng-model="style2" placeholder="Type: changeSize, changeColor or italic"><br>
<input ng-model="style3" placeholder="Type: changeSize, changeColor or italic"><br>
```

Here you can see that I have applied the CSS to the `<p>` tag using the `ng-class` but this time I have called the classes by the order they were created, `style1` is automatically bound to the first class, similarly `style2` and `style3` are bound to the second and third class.

Then I have created three Textboxes that are bound to `style1`, `2` and `3` separately, whatever CSS name is provided in these textboxes, the same CSS class will be applied to the Text provided in the `<p>` tag.

Output:



Changes Will be Seen in Me

Change Color (apply "changeColor" class)
 Change Size (apply "changeSize" class)
 Make it Italic (apply "italic" class)

I Will Show Second Method

Type: changeSize change

I Will Show Third Method

Type: changeSize, chang
Type: changeSize, chang
Type: changeSize, chang

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <style>
        .changeColor {
            color:blue;
        }
        .changeSize {
            font-size:30px;
        }
        .italic {
            font-style:italic;
        }
    </style>
</head>
<body>
    <div ng-controller="ctrl">
        Changes Will be Seen in Me
        <ul style="list-style-type: none;">
            <li><input type="checkbox"/> Change Color (apply "changeColor" class)</li>
            <li><input type="checkbox"/> Change Size (apply "changeSize" class)</li>
            <li><input type="checkbox"/> Make it Italic (apply "italic" class)</li>
        </ul>
        I Will Show Second Method
        <input type="text" value="Type: changeSize change" />
        I Will Show Third Method
        <input type="text" value="Type: changeSize, chang" />
        <input type="text" value="Type: changeSize, chang" />
        <input type="text" value="Type: changeSize, chang" />
    </div>
</body>
</html>
```

```
        }
    </style>
</head>
<body>
<p ng-
class="{changeColor: Color, changeSize: Size, italic: Italic}">Changes Will be Seen in Me</p>
<input type="checkbox" ng-model="Color"> Change Color (apply "changeColor" class)<br>
<input type="checkbox" ng-model="Size"> Change Size (apply "changeSize" class)<br>
<input type="checkbox" ng-model="Italic"> Make it Italic (apply "italic" class)
<hr>
<p ng-class="style">I Will Show Second Method</p>
<input type="text" ng-model="style" placeholder="Type: changeSize changeColor italic">
<hr>
<p ng-class="[stle1, stle2, stle3]">I Will Show Third Method</p>
<input ng-model="stle1" placeholder="Type: changeSize, changeColor or italic"><br>
<input ng-model="stle2" placeholder="Type: changeSize, changeColor or italic"><br>
<input ng-model="stle3" placeholder="Type: changeSize, changeColor or italic"><br>
</body>
</html>
```

Determine Whether User Has Used Cut, Copy or Paste For Data Using AngularJS

Introduction

This explains the ng-cut, ng-copy and ng-paste directives of AngularJS.

Using AngularJS you can determine whether the user has cut, copied or pasted anything on the WebPage using the directives provided by the AngularJS.

It might be possible that you don't want the end user to copy, cut or paste anything new on your webpage, for example we don't want anyone to copy our content from our article, in such a cases we can apply AngularJS and determine whether something is copied or not and can take action accordingly, this article will help you to apply such a functionalities to your WebPage.

I will first explain how to determine whether something is copied or not.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application as in the following:

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

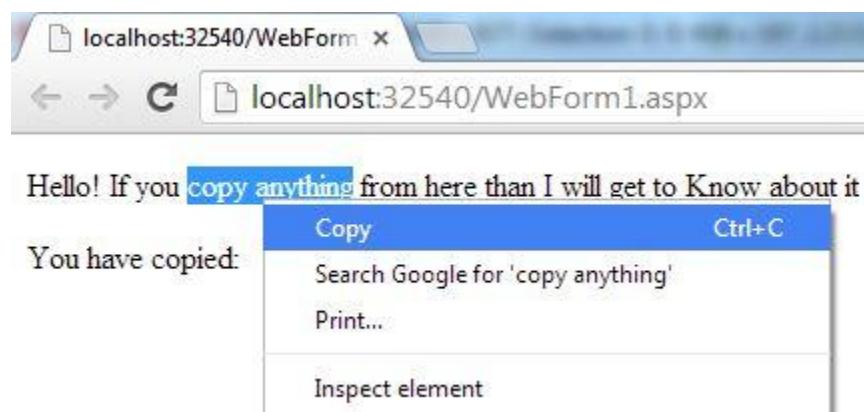
Step 2 - Now I will work on the ViewModel as in the following:

```
<body>
    <p ng-copy="copied=true" ng-model="value"> Hello!
        If you copy anything from here
        than I will get to Know about it
    </p>
    <p>You have copied: <b>{{copied}}</b></p>
</body>
```

For finding the copy we need to use the ng-copy directive, here I have used a <p> tag in which some text is written.

Then one more <p> tag is used to show True as the user will copy whether from the mouse or from the keyboard.

Output: On running the application I select some text and right-click to copy it.



As I select the "copy", you can see that "True" is shown in the output.



Now I will show you the "ng-paste" directive, which will show whether the user has pasted in anything or not.

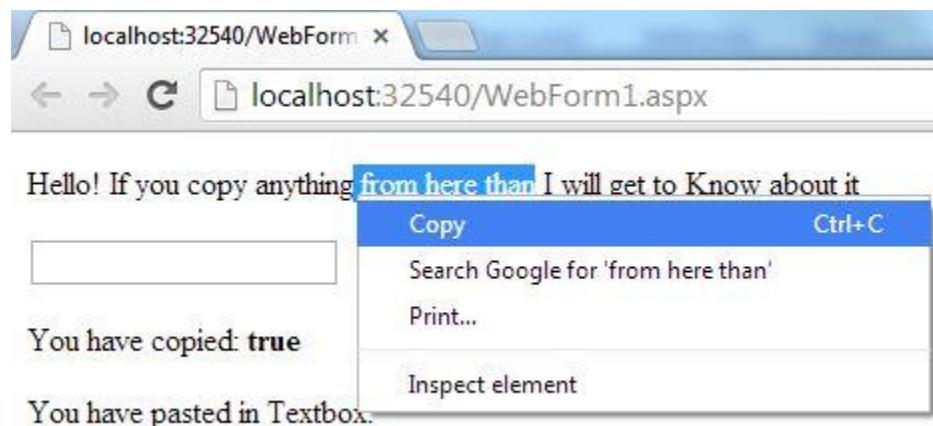
Step 3 - Modify your View Model with this code:

```
<body>
  <p ng-copy="copied=true" ng-model="value"> Hello!
    If you copy anything from here
    than I will get to Know about it
  </p>
  <input ng-paste="paste=true" type="text">
  <p>You have copied: <b>{{copied}}</b></p>
  <p>You have pasted in TextBox: <b>{{paste}}</b></p>
</body>
```

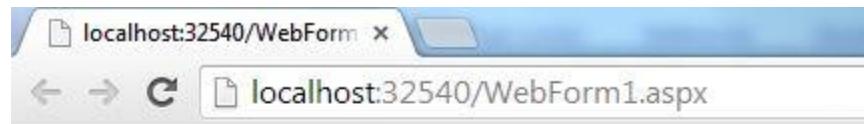
Now I had added a TextBox which is bound with ng-paste, if anything is pasted in this TextBox than True will be shown.

A `<p>` tag is used which will show the True value for paste.

Output: Now I again copy some text-



As I paste the Text in the TextBox, True is shown for Paste as well.



Hello! If you copy anything from here than I will get to Know about it

from here than

You have copied: **true**

You have pasted in Textbox: **true**

Now I will show you the **ng-cut** Directive, which will show whether user have cut anything or not.

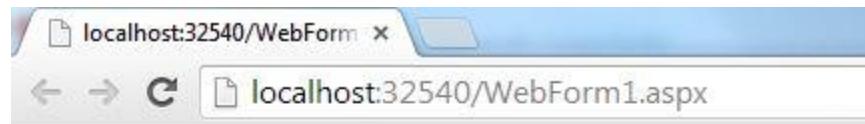
Step 3 - Modify your View Model with this code:

```
<body>
  <p ng-copy="copied=true" ng-model="value"> Hello!
    If you copy anything from here
    than I will get to Know about it
  </p>
  <input ng-cut="cut=true" ng-paste="paste=true">
  <p>You have copied: <b>{{copied}}</b></p>
  <p>You have cut from TextBox: <b>{{cut}}</b></p>
  <p>You have pasted in TextBox: <b>{{paste}}</b></p>
</body>
```

In the TextBox I have added ng-cut, this will bind the TextBox with the Cut Event.

Then I took one more <p> tag which will show the True value whenever the Text will be cut in the TextBox.

Output: Now I will first copy the text and paste it into the TextBox.



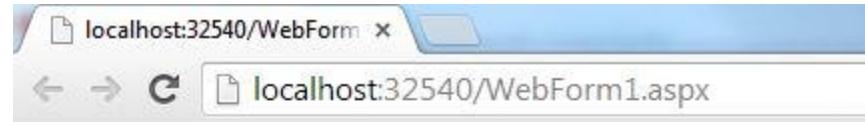
Hello! If you copy anything from here than I will get to Know about it

You have copied: **true**

You have cut from Textbox:

You have pasted in Textbox: **true**

Now as I press Ctrl+X or right-click on the text and cut it than you can see that Cut has become True.



Hello! If you copy anything from here than I will get to Know about it

You have copied: **true**

You have cut from Textbox: **true**

You have pasted in Textbox: **true**

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
<body>
    <p ng-copy="copied=true" ng-model="value"> Hello!
        If you copy anything from here
        than I will get to Know about it
    </p>
    <input ng-cut="cut=true" ng-paste="paste=true">
    <p>You have copied: <b>{{copied}}</b></p>
    <p>You have cut from Textbox: <b>{{cut}}</b></p>
    <p>You have pasted in Textbox: <b>{{paste}}</b></p>
</body>
</html>
```

ng-disabled Directive of AngularJS

Introduction

This explains the ng-disabled directive of AngularJS.

In many websites you must have seen that if your permanent address is also your current address then you just need to provide one of them and the other becomes disabled. This article will help you to create a similar application using the AngularJS ng-disabled directive.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the ViewModel.

In the View Model I am taking two Text Areas and one Checkbox, the first Text Area will be for the Permanent Address and the second is for the Current Address, if the user's Permanent Address and Current Address are the same then the user will check the checkbox then in that case the second Text Area will become disabled.

Let's see the code for this:

```
<body>
  <form id="form1" runat="server">
    <table>
      <tr>
        <td>
          <label>Your Permanent Address:</label>
        </td>
        <td>
          <textarea placeholder="Click Here"></textarea>
        </td>
      </tr>
      <tr>
        <td>
          <label>Your Permanent Address is your Current Address than Click Me: </label>
        </td>
        <td>
          <input type="checkbox" />
        </td>
      </tr>
      <tr>
        <td>
          <label>Your Current Address Address:</label>
        </td>
        <td>
          <textarea placeholder="Click Here"></textarea>
        </td>
      </tr>
    </table>
  </form>
</body>
```

Here I have created a table in which two Text Areas and a checkbox is used but nothing special is done, if at this time we run the application then we will get output like this.

Output: Here you can see that I have provided the address in both the Text Area, even the checkbox was checked.



The screenshot shows a web page with the URL `localhost:25780/WebForm1.aspx`. It contains two text areas and one checkbox. Both text areas have the same value: "Shastri Nagar Ghaziabad". The checkbox is checked.

Your Permanent Address:	Shastri Nagar Ghaziabad
Your Permanent Address is your Current Address than Click Me:	<input checked="" type="checkbox"/>
Your Current Address Address:	Shastri Nagar Ghaziabad

This was not our intent; we need an application in which one TextArea is disabled when the same value is entered.

Step 3 - Now you need to modify your code with this one:

```
<body>
<form id="form1" runat="server">
<table>
<tr>
<td>
<label>Your Permanent Address:</label>
</td>
<td>
<textarea placeholder="Click Here"></textarea>
</td>
</tr>
<tr>
<td>
<label>Your Permanent Address is your Current Address than Click Me: </label>
</td>
<td>
<input type="checkbox" ng-model="check" />
</td>
</tr>
<tr>
```

```

<td>
    <label>Your Current Address Address:</label>
</td>
<td>
    <textarea ng-disabled="check" placeholder="Click Here"></textarea>
</td>
</tr>
</table>
</form>
</body>

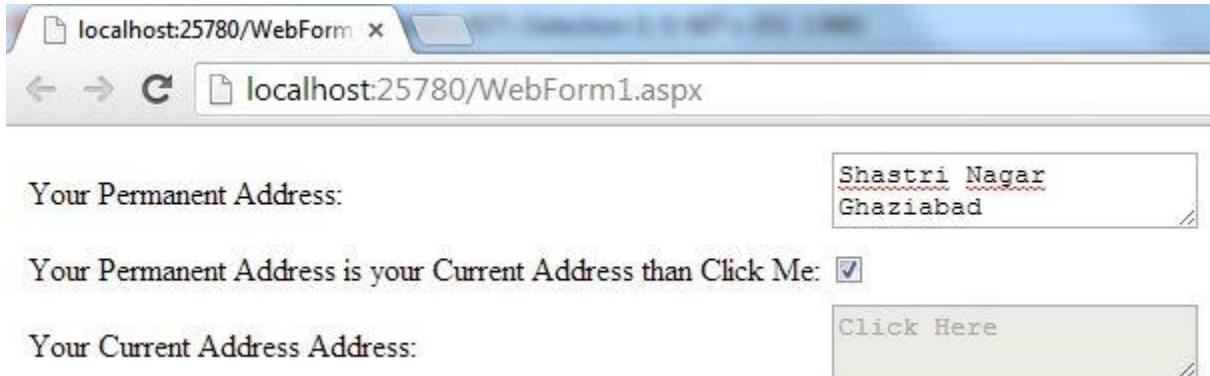
```

Now I have bound the checkbox using the ng-model. Towards the end the second Text Area is bound to the checkbox's check using the ng-disabled.

This second Text Area and Checkbox are bound so that if the checkbox is checked then the Text Area will be disabled and if it's unchecked then the Text Area will remain enabled.

Now I will run my application and you will see that the desired result is produced.

Output:



Your Permanent Address: Shastri Nagar
Ghaziabad

Your Permanent Address is your Current Address than Click Me:

Your Current Address Address: Click Here

Now you can see that first I have provided my first address and then I checked the check box, on checking the checkbox the second Textarea is disabled and I was not allowed to enter anything.

ngClassOdd and ngClassEven Directives of AngularJS

Introduction

This explains the ngClassOdd and ngClassEven directives of AngularJS.

AngularJS provide a feature by which you can apply CSS just like that of a Grid, in other words in Odd and Even order. In this article I will show you two directives that can be used together to show even and odd data in a different manner.

First I will show you the "ngClassOdd" directive.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now I will create a small CSS Class in the Head Section of my application.

```
<style>
    .classOdd {
        color: blue;
        font-size: 30px
    }
</style>
```

You can see that I have created a small CSS class in which the color and font size are declared

Step 3 - Now I will work on the View Model, in other words the design part of this application. Here I am creating a simple list of names by using ng-init and then display them using the ng-repeat. I have already explained ng-init and ng-repeat in the : **ng-init and ng-repeat Directive of AngularJS**.

```
<body>
  <div ng-init="cars=[{name:'Anu',car:'i20'},
    {name:'Anubhav',car:'Verna'},
    {name:'Mohit',car:'Audi A6'}]">
    <ul>
      <li ng-repeat="name in cars">
        <p ng-class-odd=""classOdd"">{{name.name}} likes {{name.car}}</p>
      </li>
    </ul>
  </div>
</body>
```

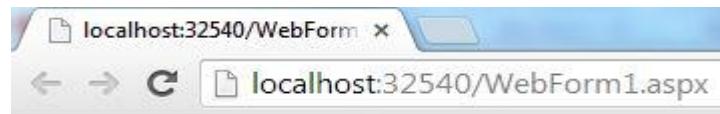
Here I used a Div in which some values are initialized using the ng-init.

Then I used a List I created in which ng-repeat is used, ng-repeat helps us to show the data provided in ng-init, the name in "name in cars" is a variable that can be changed.

Then a <p> tag is used in which the ng-class-odd directive is used, in this directive a CSS class is provided that was created in the head section. Data will be shown using this <p> tag, for showing the Data binding is done in this format: {{variable name.initialized variable}}.

Now our application is ready for execution executed so let's see the output.

Output:



- Anu likes i20
- Anubhav likes Verna
- Mohit likes Audi A6

You can see that the odd text is in the Blue color and has a greater font size.

Now I will work on the "ngClassEven".

Step 1 - I will create one more CSS class in the Style tag in the Head Section that will be used by ngClickEven. So, now the Script tag is modified to:

```
<style>
.classEven {
    color: red;
    font-size:40px
}
.classOdd {
    color: blue;
    font-size:30px
}
</style>
```

You can see that classEven is added in which a different font color and a different font size are declared, so now the Even Data should be seen in the color Red and in a bigger size than the odd data.

Step 2 - Our ViewModel also must be modified, so now the ViewModel will look like this:

```
<body>
  <div ng-init="cars=[{name:'Anu',car:'i20'},
    {name:'Anubhav',car:'Verna'},
    {name:'Mohit',car:'Audi A6'}]">
    <ul>
      <li ng-repeat="name in cars">
        <p ng-class-odd=""classOdd"" ng-class-
even=""classEven"">{{name.name}} likes {{name.car}}</p>
      </li>
    </ul>
  </div>
</body>
```

Here everything is the same except that in the `<p>` tag `ng-class-even` is used, in `ng-class-odd` the new CSS class is called.

Now our application is ready to be executed.

Output:



The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
  <script src="angular.min.js"></script>
```

```
<style>
    .classEven {
        color: red;
        font-size:40px
    }
    .classOdd {
        color: blue;
        font-size:30px
    }
</style>
</head>
<body>
    <div ng-init="cars=[{name:'Anu',car:'i20'},
        {name:'Anubhav',car:'Verna'},
        {name:'Mohit',car:'Audi A6'}]">
        <ul>
            <li ng-repeat="name in cars">
                <p ng-class-odd="'classOdd'" ng-class-
even="'classEven'">{{name.name}} likes {{name.car}}</p>
            </li>
        </ul>
    </div>
</body>
</html>
```

ng-show Directive of AngularJS

Introduction

This will tell you about the ng-show directive of AngularJS.

In many WebSites you have certainly seen that if your permanent address is different from your current address then you get a new text area for providing your current or mailing address as well. This article will help you to create a similar application using the AngularJS ng-show directive.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file now you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the ViewModel.

```
<body>
    <form id="form1" runat="server">
        <table>
            <tr>
                <td>
                    <label>Your Permanent Address:</label>
                </td>
                <td>
```

```
<textarea placeholder="Click Here"></textarea>
</td>
</tr>
<tr>
<td>
    <label>If Your Permanent Address is not Your Current Address then Click Me: </label>
</td>
<td>
    <input type="checkbox" />
</td>
</tr>
<tr>
<td>
    <label>Your Current Address:</label>
</td>
<td>
    <textarea placeholder="Click Here"></textarea>
</td>
</tr>
</table>
</form>
</body>
```

Here I have created a table in which two Text Areas and a checkbox is used but nothing special is done, if at this time we run the application then we will get an output like this one:

Output: Here you can see that both of the Text Areas are available and it doesn't matter whether you have the same mailing and permanent address or not.

localhost:25780/WebForm x

localhost:25780/WebForm1.aspx

Your Permanent Address:	<input type="text" value="Shastri Nagar
Ghaziabad"/>
Your Permanent Address is your Current Address than Click Me:	<input checked="" type="checkbox"/>
Your Current Address Address:	<input type="text" value="Shastri Nagar
Ghaziabad"/>

This was not our moto; we need an application in which a New Textarea should only be provided in those cases where the mailing address is different from the permanent address.

Step 3 - Now you need to modify your code with this one:

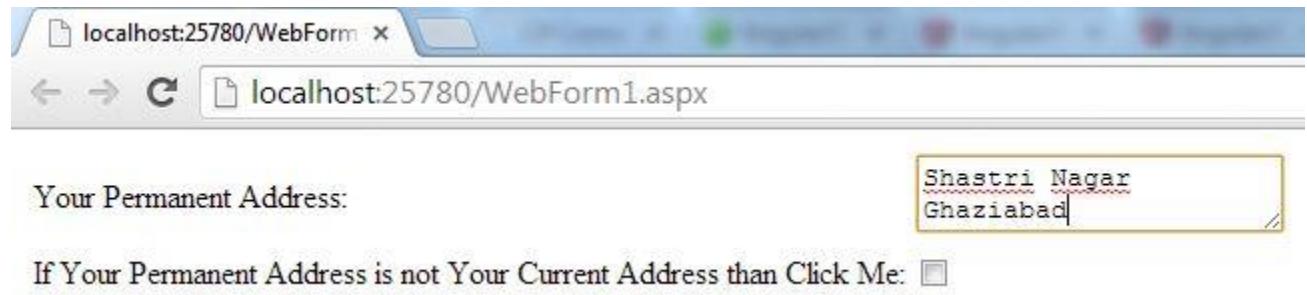
```
<body>
<form id="form1" runat="server">
<table>
<tr>
<td>
<label>Your Permanent Address:</label>
</td>
<td>
<textarea placeholder="Click Here"></textarea>
</td>
</tr>
<tr>
<td>
<label>If Your Permanent Address is not Your Current Address then Click Me:</label>
</td>
<td>
<input type="checkbox" ng-model="check" />
</td>
</tr>
```

```
<tr ng-show="check">
  <td>
    <label>Your Current Address:</label>
  </td>
  <td>
    <textarea placeholder="Click Here"></textarea>
  </td>
</tr>
</table>
</form>
</body>
```

Now I have bound the checkbox using ng-model. After this a new table row is created that is bound to the check of checkbox using the ng-show. What this will do is this row and its components will only be seen when the checkbox is checked.

Now I will run my application and you will see that the desired output is produced.

Output:



Now you can see that only one Textarea is available to the user, the checkbox is unchecked so that's why the second TextBox is not shown to the user.

ng-hide Directive of AngularJS

Introduction

This explains the ng-hide directive of AngularJS.

In many web sites you have certainly seen that if your permanent address is similar to your current address then one of the Textareas is either disabled or is not shown to the user. This article will help you to create a similar application using the AngularJS ng-hide directive.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file now you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the ViewModel.

```
<body>
    <form id="form1" runat="server">
        <table>
            <tr>
                <td>
                    <label>Your Permanent Address:</label>
                </td>
                <td>
                    <textarea placeholder="Click Here"></textarea>
                </td>
            </tr>
        </table>
    </form>
</body>
```

```
</td>
</tr>
<tr>
  <td>
    <label>If Your Permanent Address is also Your Current Address then Click Me: </label>
  </td>
  <td>
    <input type="checkbox" ng-model="check" />
  </td>
</tr>
<tr ng-hide="check">
  <td>
    <label>Your Current Address:</label>
  </td>
  <td>
    <textarea placeholder="Click Here"></textarea>
  </td>
</tr>
</table>
</form>
</body>
```

Here I have created a table in which two Text areas and a checkbox is used but nothing special is done, if at this time we run the application then we will get output like the following.

Output: Here you can see that both of the Text Areas are available and it doesn't matter whether you have the same mailing and permanent address or not.



The screenshot shows a web page titled "localhost:25780/WebForm1.aspx". It contains two textareas. The first textarea is labeled "Your Permanent Address:" and contains the text "Shastri Nagar Ghaziabad". The second textarea is labeled "Your Current Address Address:" and also contains the text "Shastri Nagar Ghaziabad". A checkbox labeled "Your Permanent Address is your Current Address than Click Me:" is checked.

This was not our moto; we need an application in which one of the Textareas is hidden if both of the Addresses are the same.

Step 3 - Now you need to modify your code with this code:

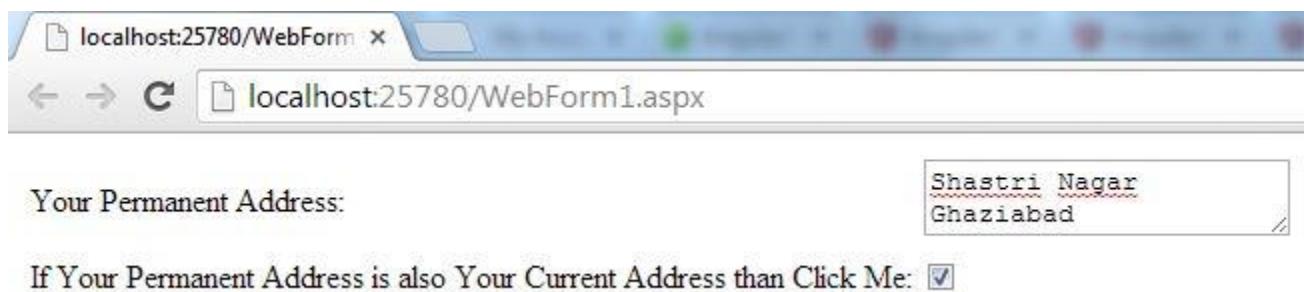
```
<body>
    <form id="form1" runat="server">
        <table>
            <tr>
                <td>
                    <label>Your Permanent Address:</label>
                </td>
                <td>
                    <textarea placeholder="Click Here"></textarea>
                </td>
            </tr>
            <tr>
                <td>
                    <label>If Your Permanent Address is also Your Current Address then Click Me:</label>
                </td>
                <td>
                    <input type="checkbox" ng-model="check" />
                </td>
            </tr>
            <tr ng-hide="check">
                <td>
```

```
<label>Your Current Address:</label>
</td>
<td>
    <textarea placeholder="Click Here"></textarea>
</td>
</tr>
</table>
</form>
</body>
```

Now I have bound the checkbox using the ng-model. After this a new Table Row is created that is bound to the check of the checkbox using ng-hide. What this will do is this row and its components will be hidden when the checkbox is checked.

Now I will run my application and you will see that the desired output is produced.

Output:



Now you can see that first I provided my first address and then I checked the check box, on checking the checkbox the second Textarea is hidden so I can't provide any other address in the second textarea.

ng-dblclick Directive of AngularJS

Introduction

This explains the ng-dblclick directive of AngularJS.

Today's article will help you to create an application where you may use the double-click event, this article will help you to create a double-click event using AngularJS.

Step 1 - You need to first add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JavaScript file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

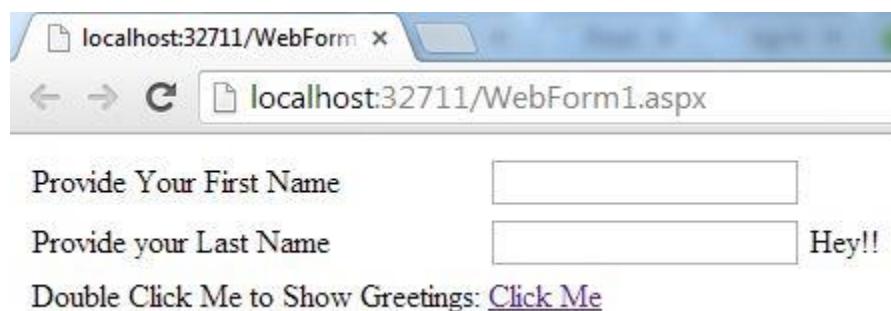
I will now create two Textboxes, one anchor and one Label. Double-clicking the anchor will replace the second TextBox with the Label. Right now I will just create the application without having any type of functionality.

```
<body>
    <form id="form1" runat="server">
        <table >
            <tr>
                <td>
                    <label>Provide Your First Name</label>
                </td>
                <td>
```

```
<input type="text" />
</td>
</tr>
<tr>
<td>
    <label>Provide your Last Name</label>
</td>
<td>
    <input type="text" />
    <label>Hey!!</label>
</td>
</tr>
<tr>
<td>
    <label>Double Click Me to Show Greetings:</label>
</td>
<td>
    <a href="">Click Me</a>
</td>
</tr>
</table>
</form>
</body>
```

If we run the application at this level then this type of output will be seen:

Output:



We don't require this type of output, this is simply a table in which a few HTML Controls are used and nothing more than that.

Now I will add functionality to this application.

Step 3 - First of all I am creating a JavaScript function in the head tag that will have some initial values to be applied to the TextBox so that it can show a default name every time it is executed.

```
<script>
    function x($scope) {
        $scope.initial = 'Anu';
    }
</script>
```

Now I will modify the code that was provided in Step 2 so that the functionality can be added to it.

```
<body>
    <form id="form1" runat="server">
        <table ng-controller="x">
            <tr>
                <td>
                    <label>Provide Your First Name</label>
                </td>
                <td>
                    <input ng-model="initial" />
                </td>
            </tr>
            <tr>
                <td>
                    <label>Provide your Last Name</label>
                </td>
                <td>
                    <input ng-hide="check" />
                    <label ng-show="check" >{{check}}</label>
                </td>
            </tr>
            <tr ng-hide="check">
```

```

<td>
    <label>Double Click Me to Show Greetings:</label>
</td>
<td>
    <a href="" ng-dblclick="check='Hello '+initial">Click Me</a>
</td>
</tr>
</table>
</form>
</body>

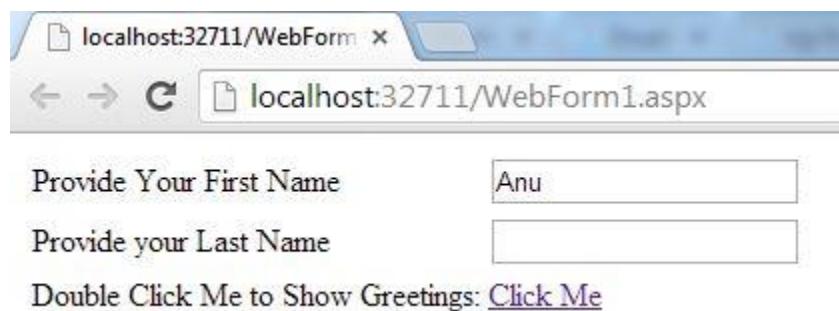
```

The function name is bound with the table using the ng-controller, in this table the first TextBox is bound with the initial value provided in the script tag, after this one more TextBox and a label are declared in the same td and both are bound to the same dblclick directive as well but only one will be seen at a time, initially the TextBox will be seen but double-clicking will hide the TextBox and will show the Label.

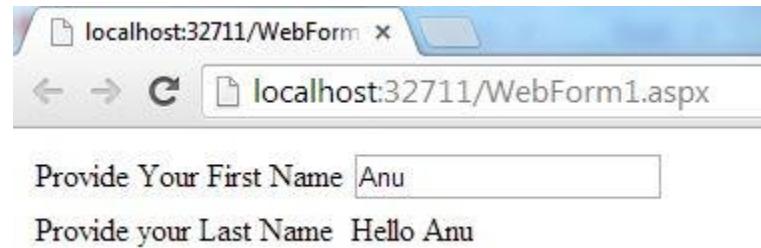
At the end an anchor is taken in that double-click event; is declared using the dblclick.

Now our application is created and is ready to be executed.

Output: On running the application you will see that the first TextBox has the value that was declared as the initial value in the JavaScript function. The second TextBox is also available but Label is nowhere to be seen.



Now as I double-click this anchor you can see that a Label is seen along with the value provided in the first TextBox.



Key Up and Key Down Event in AngularJS

Introduction

In previous I told you about:

- **ng-click and ng-blur Directive of AngularJS**
- **ng-init and ng-repeat Directive of AngularJS**
- **ng-change Directive of AngularJS**
- **ng-BindHtml Directive of AngularJS**
- **ng-bind-template directive of AngularJS.**
- **ngClassOdd and ngClassEven Directive of AngularJS**
- **ng-cut, ng-copy and ng-paste Directive of AngularJS**
- **ng-disabled Directive of AngularJS**
- **ng-show Directive of AngularJS.**
- **ng-hide Directive of AngularJS.**
- **ng-dblclick Directive of AngularJS**

This explains the ng-keyup and ng-keydown directives of AngularJS.

ng-keyup occurs when the key is released after it has been pressed but ng-keydown occurs when the key is pressed, it will not wait for it to come up, so for key up you need to press it every time you need to trigger it but key down will continue to work until you release the key after pressing it.

Step 1 - You need to first add the external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the External File you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now first I will show you the key down event, for this you need to add this code in the body section of your application:

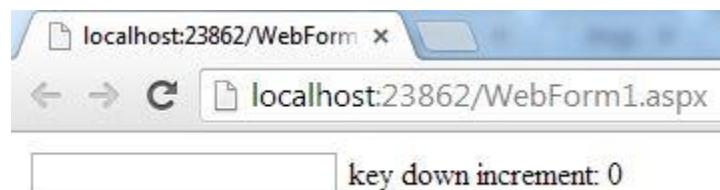
```
<body>
  <form id="form1" runat="server">
    <div>
      <input ng-keydown="increment = increment + 1" ng-init="increment=0">
      key down increment: {{increment}}
    </div>
  </form>
</body>
```

Here I had initialized the TextBox by a default value of 0, this TextBox is also bound to the key down event, and in other words this initial value will be incremented by one on each click.

After this the increment is bound with a text that will help to determine the number of increments made.

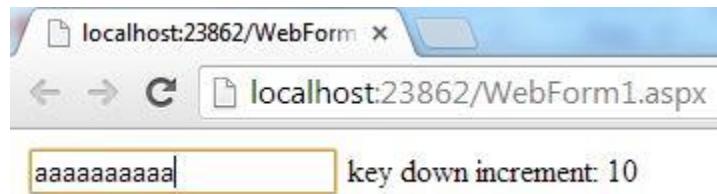
Now our application is created and we can check it.

Output: On running the application you will get an output like this one:



Here you can see that by default zero is displayed in the number of key down events.

But as I press a key for a long time you can see that the Click Counter is also increasing at a great speed:



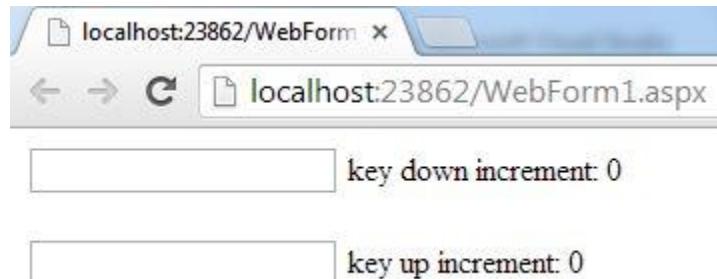
Step 3 - Now I will work on the **Key Up Event**, for this you need to modify your code with this one:

```
<input ng-keyup="upincrement = upincrement + 1" ng-init="upincrement=0">  
key up increment: {{upincrement}}
```

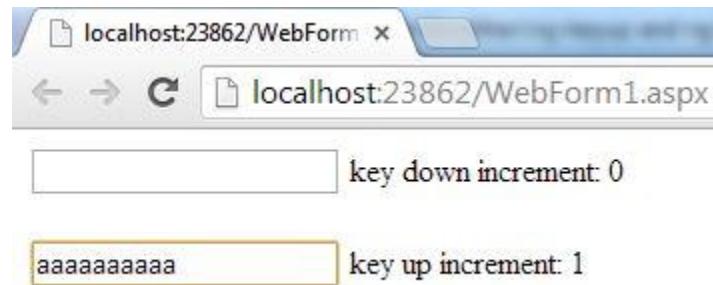
Here once again a click counter is created but this time it will work on key up, every time a key is released this counter will increase its value by one.

Let's run the application and see the output

Output: As we had seen in key down, first the counter will be at zero.



Now I am pressing a key and will not leave it for a few milliseconds, let's see what will happen.



You can see that the counter is incremented by 1 only when I leave the key after pressing it.

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <input ng-keydown="increment = increment + 1" ng-init="increment=0">
            key down increment: {{increment}}
            <br />
            <br />
            <input ng-keyup="upincrement = upincrement + 1" ng-init="upincrement=0">
            key up increment: {{upincrement}}
        </div>
    </form>
</body>
</html>
```

ng-Switch Directive of AngularJS

Introduction

This explains the ng-Switch directive of AngularJS.

ng-switch can be used to swap things; it uses two or more directives in it that are ng-switch-when and ng-switch-default, if these are not used then whatever you have provided in the scope will be provided as-is and not on the selection made. Values that are provided in the scope can be swapped using ng-switch.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application where this directive along with its supporting directives will be used.

First I will create a JavaScript function in which some values are to be passed that will be swapped by our directive. For that you need to add this code in the head section of your application:

```
<script>
    function func($scope) {
        $scope.div = ['div1', 'div2', 'div3'];
        $scope.selectedDiv = $scope.div[0];
    }
</script>
```

Here I have created a function in which some initial values are passed, first the value is selected as the default value using the div [0].

Step 3 - Now I will work on the ViewModel of this application.

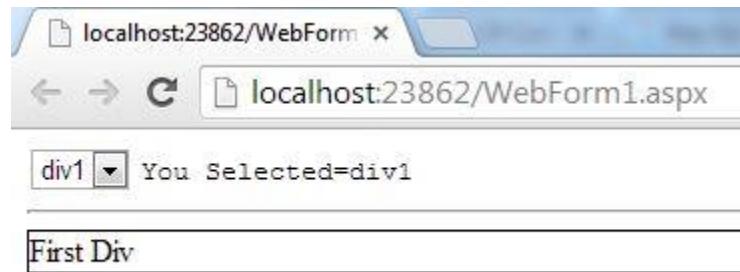
```
<div ng-controller="func">
    <select ng-model="selectedDiv" ng-options="item for item in div">
    </select>
    <tt>You Selected={{selectedDiv}}</tt>
    <hr/>
    <div class="animate-switch-container" ng-switch on="selectedDiv">
        <div style="border:1px solid black" ng-switch-when="div1">First Div</div>
        <div style="border:1px solid black" ng-switch-when="div2">Second Div</div>
        <div style="border:1px solid black" ng-switch-default>Default Div or Third Div</div>
    </div>
```

Here I have created a parent div that is bound to the controller using the ng-controller, then selectedDiv is bound to the drop down list using the ng-model.

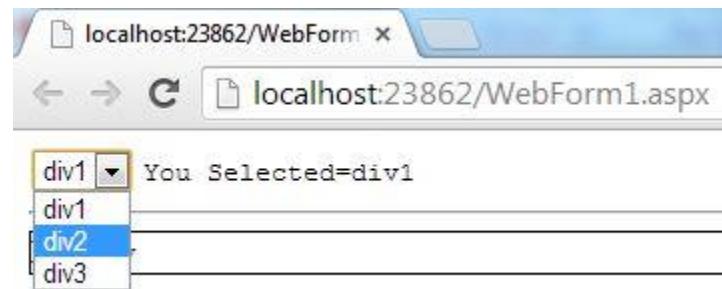
Then I have created a Div in which three Divs are available, this parent div is bound to the ng-switch, here switching is started and will be displayed here. Next three Divs are bound using ng-switch-when and ng-switch-default. The Div that are bound to ng-switch-when will be displayed only when the corresponding text is selected from the Drop Down List and Default will be displayed by default on running the application.

Now our application is created and can be executed.

Output: On running the application you will see that first the Div is selected by default in the Drop Down and corresponding Text is shown below it.



Now if I click on the Drop Down then all three Divs will be shown in it.



Now as I select the other Div, the corresponding text will again be shown below it.



The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<script src="angular.min.js"></script>
<script>
    function func($scope) {
        $scope.div = ['div1', 'div2', 'div3'];
        $scope.selectedDiv = $scope.div[0];
    }
</script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="func">
            <select ng-model="selectedDiv" ng-options="item for item in div">
            </select>
            <tt>You Selected={{selectedDiv}}</tt>
            <hr/>
            <div class="animate-switch-container" ng-switch on="selectedDiv">
                <div style="border:1px solid black" ng-switch-when="div1">First Div</div>
                <div style="border:1px solid black" ng-switch-when="div2">Second Div</div>
                <div style="border:1px solid black" ng-switch-default>Default Div or Third Div</div>
            </div>
        </form>
    </body>
</html>
```

ng-Transclude Directive of AngularJS

Introduction

This explains the ng-Transclude Directive of AngularJS.

Basically Transclusion means inclusion of a document or a part of a document in another document by passing a reference. AngularJS provides a Directive named "ngTransclude" to insert a document in the DOM Elements.

Now I will create a sample application that will help you understand this directive.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file now you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the External JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application where this directive along with it's supportive directives will be used.

First I will create a JavaScript function in which some initial values will be provided, also a directive will be created in which some data in a Div will be passed.

```
<script>
    function x($scope) {
        $scope.title = 'I am Coming from the Directive';
```

```
$scope.domDiv = 'I am provided Directly into the DOM';
angular.module('transclude', [])
.directive('p', function () {
    return {
        restrict: 'AE',
        transclude: true,
        scope: { title: '@' },
        template: '<div style="border: 2px solid red;">' +
            '<div style="color: blue">{{title}}</div>' +
            '<div ng-transclude></div>' +
            '</div>'  });
}); </script>
```

Here title and domDiv are the initial values that are passed using the scope, then the second to create the function is used, in this method the first module is created, in this module a directive is created.

In this directive "restrict" is used, by default the directive is restricted to the attribute, if we want the directive to be triggered by an element name then we need to use the restriction option. There are three types of restriction option:

1. A- It will only match the attribute Name.
2. E- It will only match the Element Name.
3. AE- It will match the Element and Attribute both.

Then I set transclude to true, at the end in the template some div are passed to which some style is bound along with the initial values that were passed a little while ago.

Step 3 - Our work on View is completed and now I will work on the View Model of this application.

```
<form id="form1" runat="server">
<div ng-app="transclude">
<div ng-controller="x">
    <input ng-model="title" style="width:200px"><br>
    <input ng-model="domDiv" style="width:200px"><br/>
    <p title="{{title}}>{{domDiv}}</p>
</div>
</div>
</form>
```

Here the first div that is the main div is bound to the ng-app, in this ng-app the module name will be provided that was created in the JavaScript function, then the second div is bound to the controller, the controller is the function name.

Then two Textboxes are created and both are bound to the initial values that were provided in Step 2. One `<p>` tag is also used in which the first initial value is provided in the Title and the second initial value is bound as Text to be shown.

But this `<p>` tag will show both the initial values in the Div format along with border and color that was provided in the directive.

Now our application is created and is ready for execution.

Output: On running the application both the initial values will be available in the Textboxes, and both of these initial values will be shown in a red bordered Div.



This shows that Transculete helps to include two different documents in one document.

Now if I make changes in any of the TextBoxes then you will see that corresponding changes are shown in the Div also.



The complete code of this application is as follows:

```

<html ng-app="transclude" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.title = 'I am Coming from the Directive';
            $scope.domDiv = 'I am provided Directly into the DOM';
        }
    
```

```

        angular.module('transclude', [])
            .directive('p', function () {
                return {
                    restrict: 'AE',
                    transclude: true,
                    scope: { title: '@' },
                    template: '<div style="border: 2px solid red;">' +
                        '<div style="color: blue">{{title}}</div>' +
                        '<div ng-transclude></div>' +
                        '</div>'
                };
            });
        </script>
    </head>
    <body>
        <form id="form1" runat="server">
            <div ng-app="transclude">
                <div ng-controller="x">
                    <input ng-model="title" style="width:200px"><br>
                    <input ng-model="domDiv" style="width:200px"><br/>
                    <p title="{{title}}>{{domDiv}}</p>
                </div>
            </div>
        </form>
    </body>
</html>

```

ng-Value Directive of AngularJS

Introduction

This explains the ng-Value directive of AngularJS.

ng-Value binds the input or checkbox control with some values, in this way it changes the bound value of ng-model with the currently selected value.

ng-Value mainly works with ng-Repeat when we are creating a dynamic List of values. Now I will create a sample application that will help you understand this directive.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application that will help you understand this directive.

First I will create a JavaScript function in which some initial values will be passed.

```
<script>
function x($scope) {
    $scope.cars = ['AUDI', 'BMW', 'MERCEDES'];
    $scope.my = { favorite: 'AUDI' };
} </script>
```

Here I passed some initial car names in the "cars", then I selected "AUDI" as the default and favorite value.

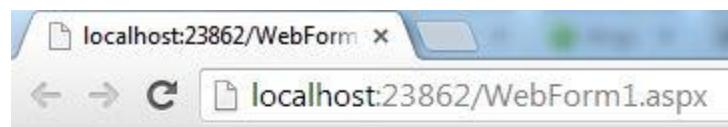
Step 3 - Our work on the View is completed and now I will work on the View Model of this application.

```
<body>
<div>
<form ng-controller="x">
<h4 id="favorite_CAR">Which is your favorite?</h4>
<label ng-repeat="car in cars" for="{{car}}">
{{car}}
<input type="radio" ng-model="my.favorite" ng-value="car">
</label>
<div>Your Favourite Car is: {{my.favorite}}</div>
</form>
</div>
</body>
```

The form is bound with the function name using the ng-controller, then a label is taken that will act as a Dynamic List, this label is bound using the ng-repeat directive.

Then a Label is taken whose type is set to checkbox, this will create dynamic checkboxes for each data that were declared in the initial values at the JavaScript function. At the end a div is created in which some labels are bound with the selected checkbox value. Now our application is ready for execution.

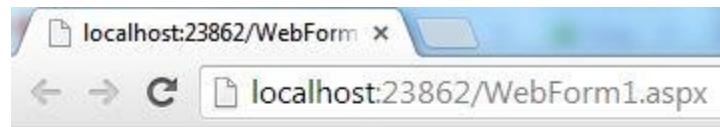
Output: On running the application you will see that AUDI will be selected as the default value as in the following:



Which is your favorite?

AUDI BMW MERCEDES
Your Favourite Car is: AUDI

This is because we had selected this value in the JavaScript function as the default value. Now I can select any other value also and you can see that regarding changes are seen in the Label also:



Which is your favorite?

AUDI BMW MERCEDES
 Your Favourite Car is: MERCEDES

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<script src="angular.min.js"></script>
<script>
  function x($scope) {
    $scope.cars = ['AUDI', 'BMW', 'MERCEDES'];
    $scope.my = { favorite: 'AUDI' };
  }
</script>
</head>
<body>
<div >
<form ng-controller="x">
<h4 id="favorite_CAR">Which is your favorite?</h4>
<label ng-repeat="car in cars" for="{{car}}">
  {{car}}
  <input type="radio" ng-model="my.favorite" ng-value="car">
</label>
<div>Your Favourite Car is: {{my.favorite}}</div>
</form>
</div>
</body></html>
```

Convert Text To Uppercase Using AngularJS

Introduction

In this I will tell you how to convert text into uppercase letters using AngularJS.

AngularJS provides a feature for converting all the letters of text into uppercase letters. I will explain this feature by creating a sample application.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application that will help you understand this filter.

First I will create a JavaScript function in which some initial values will be passed.

```
<script>
    function x($scope) {
        $scope.test = "this text will be displayed in Uppercase";
    }
</script>
```

Here I have created a function named "x", in this function the initial value is passed to a variable named "test", this text is the text that will be converted into the Uppercase Letters but as you can see that at this point all the letters are in lowercase letters.

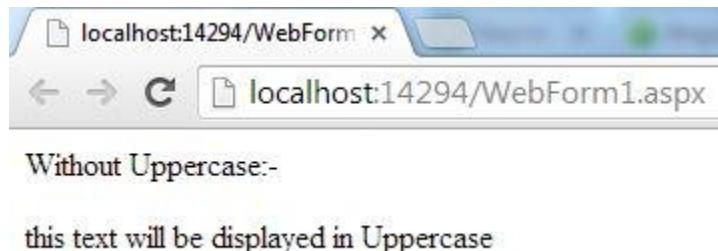
Step 3 - Our work on View is completed and now I will work on the View Model of this application.

```
<body>
<form id="form1" runat="server">
<div ng-controller="x">
Without Uppercase:- <p>{{test}}</p>
</div>
</form>
</body>
```

Here a Div is created that is bound to the function using the ng-controller.

In this Div a `<p>` tag is used that is bound to the initial value but it's not made to be converted into uppercase.

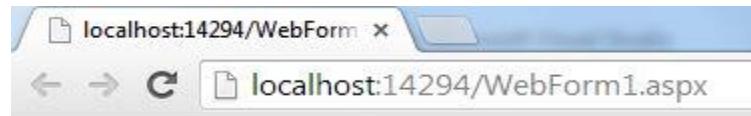
If at this time I run the application than a simple output like this one will be seen:



Nothing is converted to uppercase, that's because I have not applied anything to convert it but now I will add one more `<p>` tag and will convert the text to uppercase.

```
<body>
<form id="form1" runat="server">
<div ng-controller="x">
Without Uppercase:- <p>{{test}}</p> <br />
With Uppercase:- <p>{{test|uppercase}}</p>
</div>
</form>
</body>
```

In the second <p> tag you can see that I had again bound the initial value but with it one more thing is written, in other words uppercase, this should be written in this format: {{ uppercase_expression | uppercase }}. Now if I run the application then an output like this one can be seen:



Without Uppercase:-

this text will be displayed in Uppercase

With Uppercase:-

THIS TEXT WILL BE DISPLAYED IN UPPERCASE

The complete code of this application is as follows:

```

<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.test = "this text will be displayed in Uppercase";
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            Without Uppercase:- <p>{{test}}</p>
            <br />
            With Uppercase:- <p>{{test|uppercase}}</p>
        </div>
        </form>
    </body>
</html>

```

Apply Sorting Using AngularJS

Introduction

In this I will tell you how to do sorting using AngularJS.

Angular provides the feature named **orderBy** that can be used to sort the given data. Here I will create an application where sorting will be applied on multiple columns.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file now you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application that will help you understand this filter.

First I will create a JavaScript function in which some initial values will be passed.

```
<script>
function x($scope) {
    $scope.cars =
        [{ name: 'Alto 800', price: '3.5 Lakh', version: 'Manual' },
         { name: 'Eon', price: '4.2 Lakh', version: 'Manual' },
         { name: 'i20', price: '7 Lakh', version: 'Automattic' },
         { name: 'Honda City', price: '11 Lakh', version: 'Automattic' },
         { name: 'i10', price: '5 Lakh', version: 'Manual' }]
```

```

$scope.sorting = '-version';
}
</script>

```

Here I have created an Array of some data, three columns are created in which multiple names, prices and versions of car are defined.

Then reverse sorting is applied on the version of car, this is the initial sorting, later in this article I will show you how to apply sorting on all the columns.

Step 3 - Our work on the View is completed and now I will work on the View Model of this application.

```

<body>
<div ng-app>
<div ng-controller="x">
[ <a href="" ng-click="sorting=''">Default</a> ]
<table class="car">
<tr>
<th><a href="" ng-click="sorting = 'name'; reverse=!reverse">Name</a></th>
<th><a href="" ng-click="sorting = 'price'; reverse=!reverse">Price</a></th>
<th><a href="" ng-click="sorting = 'version'; reverse=!reverse">Version</a></th>
</tr>
<tr ng-repeat="car in cars | orderBy:sorting:reverse">
<td>{{car.name}}</td>
<td>{{car.price}}</td>
<td>{{car.version}}</td>
</tr>
</table>
</div>
</div>
</body>

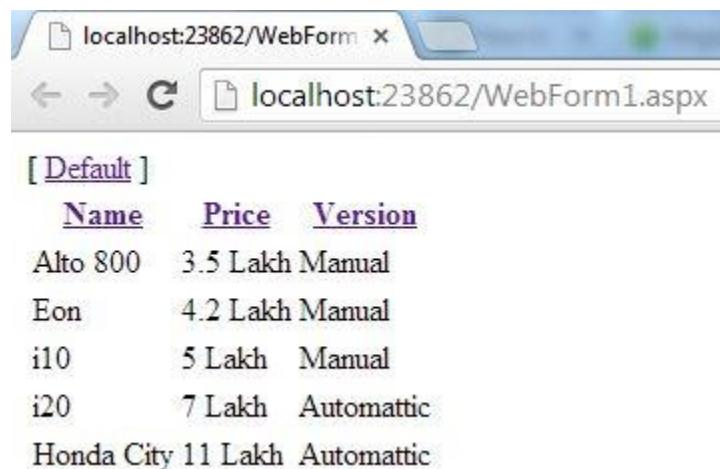
```

"x", which was the name of the function in Step 2, is bound with one of the parent Divs using ng-controller. Then a table is created in which three headings are provided in separate columns, in these headings anchors are used whose click is bound to the name or price or version, these clicks will work for sorting the data.

Then one more table is created in which the ng-repeat directive is used and its columns are bound to the name, price and version of car. As the repeat directive is used it will cerate a dynamic table of the data. In this table orderBy is used that shows that it's column will be sorted whenever "sorting" is called.

Now our application is created and is ready to for execution.

Output: On running the application you will get an output like this one:

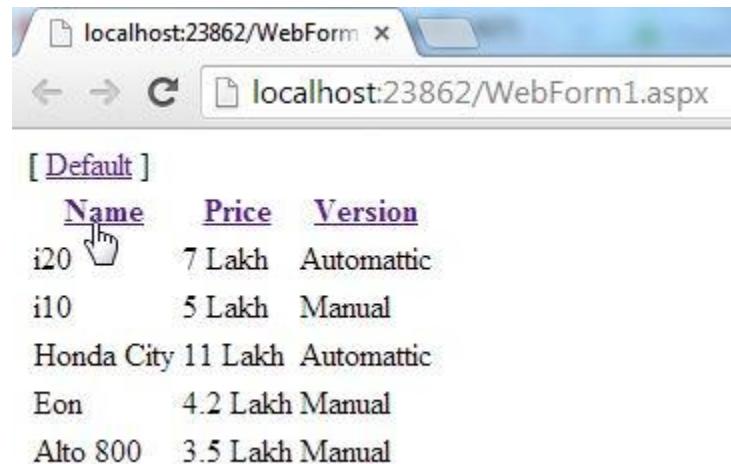


A screenshot of a Microsoft Internet Explorer browser window. The address bar shows 'localhost:23862/WebForm1.aspx'. The page content starts with '[Default]' in purple. Below it is a table with three columns: 'Name', 'Price', and 'Version'. The data rows are: 'Alto 800 3.5 Lakh Manual', 'Eon 4.2 Lakh Manual', 'i10 5 Lakh Manual', 'i20 7 Lakh Automattic', and 'Honda City 11 Lakh Automattic'.

Name	Price	Version
Alto 800	3.5 Lakh	Manual
Eon	4.2 Lakh	Manual
i10	5 Lakh	Manual
i20	7 Lakh	Automattic
Honda City	11 Lakh	Automattic

You can see that all the data is displayed in table format and in the heading section an anchor is provided whose click should provide the sorting on a specified column.

Now I will click on one of the Headers and you will see that reverse sorting is applied.

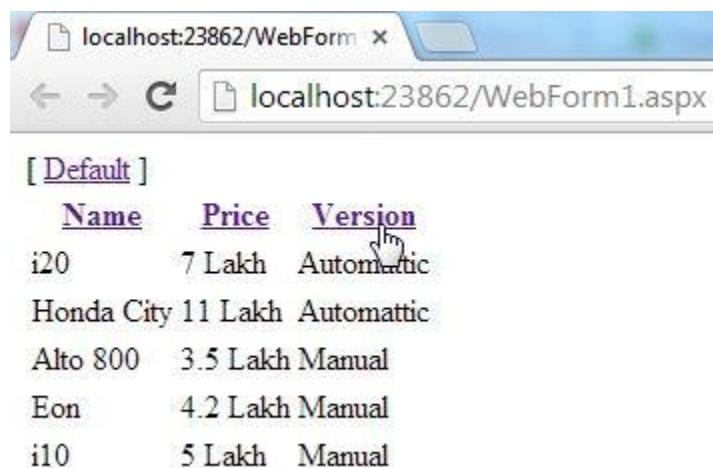


A screenshot of a web browser window titled "localhost:23862/WebForm". The address bar shows "localhost:23862/WebForm1.aspx". The page content is a table with the following data:

<u>Name</u>	<u>Price</u>	<u>Version</u>
i20	7 Lakh	Automatic
i10	5 Lakh	Manual
Honda City	11 Lakh	Automatic
Eon	4.2 Lakh	Manual
Alto 800	3.5 Lakh	Manual

This is because by default reverse sorting was made available in the JavaScript.

Now I will click on one of the Headers and again sorting will be done.



A screenshot of a web browser window titled "localhost:23862/WebForm". The address bar shows "localhost:23862/WebForm1.aspx". The page content is a table with the following data, showing rows sorted by Name:

<u>Name</u>	<u>Price</u>	<u>Version</u>
i20	7 Lakh	Automatic
Honda City	11 Lakh	Automatic
Alto 800	3.5 Lakh	Manual
Eon	4.2 Lakh	Manual
i10	5 Lakh	Manual

Now if I click on the Default Anchor then everything will be reset to it's default position.

localhost:23862/WebForm1.aspx

[Default]		
Name	Price	Version
Alto 800	3.5 Lakh	Manual
Eon	4.2 Lakh	Manual
i10	5 Lakh	Manual
i20	7 Lakh	Automattic
Honda City	11 Lakh	Automattic

The complete code of this application is as follows:

```

<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.cars =
                [{ name: 'Alto 800', price: '3.5 Lakh', version: 'Manual' },
                 { name: 'Eon', price: '4.2 Lakh', version: 'Manual' },
                 { name: 'i20', price: '7 Lakh', version: 'Automattic' },
                 { name: 'Honda City', price: '11 Lakh', version: 'Automattic' },
                 { name: 'i10', price: '5 Lakh', version: 'Manual' }]
            $scope.sorting = '-version';
        }
    </script>
</head>
<body>
    <div ng-app>
        <div ng-controller="x">
            [ <a href="" ng-click="sorting=''">Default</a> ]
            <table class="car">
                <tr>
                    <th><a href="" ng-click="sorting = 'name'; reverse=!reverse">Name</a>

```

```
<th><a href="" ng-click="sorting = 'price'; reverse=!reverse">Price</a></th>
<th><a href="" ng-click="sorting = 'version'; reverse=!reverse">Version</a></th>
</tr>
<tr ng-repeat="car in cars | orderBy:sorting:reverse">
  <td>{{car.name}}</td>
  <td>{{car.price}}</td>
  <td>{{car.version}}</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

Convert Text To Lowercase Letters Using AngularJS

Introduction

In this I will tell you how to convert text into lowercase letters using AngularJS.

AngularJS provides a feature by which you can convert all the letters of some text into uppercase letters or uppercase letters can be converted into lowercase letters. I will explain this with a sample application.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application as in the following:

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the External JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application that will help you understand this feature.

First I will create a JavaScript function in which some initial values will be passed.

```
<script>
    function x($scope) {
        $scope.test = "this text will be displayed in Uppercase";
        $scope.test1 = "THIS TEXT WILL BE DISPLAYED IN LOWERCASE";
    }
</script>
```

Here I have created a function named "x". In that function some initial values are passed to variables named "test" and "test1", the text provided in the "text" will be converted into uppercase letters but as you can see, at this point all the letters are in lowercase letters.

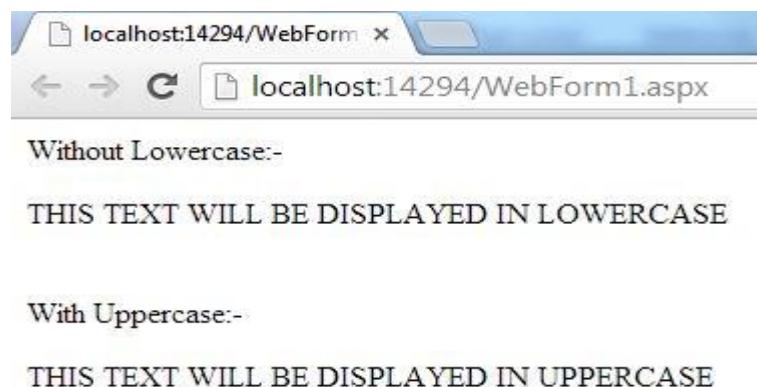
Similarly text provided in the "text1" will be converted into lowercase letters but right now they are in capital letters.

Step 3 - Our work on the View is complete. Now I will work on the View Model of this application.

```
<body>
<form id="form1" runat="server">
<div ng-controller="x">
Without Lowercase:- <p>{{test1}}</p>
<br />
With Uppercase:- <p>{{test|uppercase}}</p>
</div>
</form>
</body>
```

Here a Div is created that is bound to the function using the ng-controller.

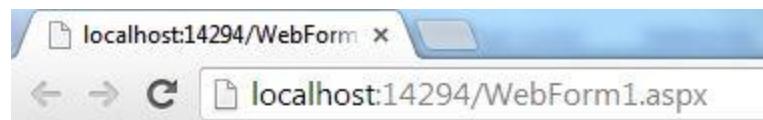
In this Div two `<p>` tags are taken, the first one is bound to the initial value of test1 but right now nothing special is done so it will not convert the data into lowercase. The second `<p>` tag is bound to the "test" but along with it "uppercase" is also written so it will convert the text into the uppercase letters. If at this time I run the application then output like this will be seen:



Nothing is converted to lowercase, that's because I have not applied anything to convert it but now I will add one more `<p>` tag and will convert the text to lowercase.

```
<body>
  <form id="form1" runat="server">
    <div ng-controller="x">
      Without Lowercase:- <p>{{test1}}</p>
      <br />
      With Uppercase:- <p>{{test|uppercase}}</p>
      <br />
      With Lowercase:- <p>{{test1|lowercase}}</p>
    </div>
  </form>
</body>
```

In the third `<p>` tag you can see that I had again bound the initial value but with it one more thing is written, lowercase, this should be written in this format: `{} lowercase_expression | lowercase{}`. Now if I run the application then output like this can be seen:



Without Lowercase:-

THIS TEXT WILL BE DISPLAYED IN LOWERCASE

With Uppercase:-

THIS TEXT WILL BE DISPLAYED IN UPPERCASE

With Lowercase:-

this text will be displayed in lowercase

Complete Code

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.test = "this text will be displayed in Uppercase";
            $scope.test1 = "THIS TEXT WILL BE DISPLAYED IN LOWERCASE";
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            Without Lowercase:- <p>{{test1}}</p>
            <br />
            With Uppercase:- <p>{{test|uppercase}}</p>
            <br />
            With Lowercase:- <p>{{test1|lowercase}}</p>
        </div>
    </form>
</body>
</html>
```

Apply "Go to Bottom" and "Go Up" on Click of Anchors

Introduction

In this I will tell you how to apply "Go to Bottom" and "Go Up" on a click of Anchors in an application.

We often we create single page applications that are in great demand at the present time, in these applications you might need a functionality by which the user can go to a different part of a page just by clicking on the link and not by scrolling. To do this kind of functionality you can use AngularJS.

Now I will create an application that will help you to create such an application.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application that will help you understand this feature.

First I will create a JavaScript function; this is the heart of this article. All the things will be described in this function.

```
<script>
  function x($scope, $location, $anchorScroll) {
    $scope.goDown = function () {
      $location.hash('down');
      $anchorScroll();
    }
    $scope.goUp = function () {
      $location.hash('up');
      $anchorScroll();
    }
  }
</script>
```

Here I have created a function named "x", but today in the function \$location and \$anchorScroll are also used along with \$scope. \$location service parses the browser address bar and make the URL available to your application.

Then I created a function that will help the user to go down just on the click of a link, this function is named "goDown". In this function a hash method is used with location, you can't change the "hash" method with any other variable name, it's fixed, in this has the method id of control that is passed to be followed on the click of the anchor.

Similarly I have created a function that will help the user to go up, here also the hash method is used and in this method the name of the control is passed that is to be followed when clicking the anchor.

Step 3 - Our work on the View is completed and now I will work on the View Model of this application.

```
<body>
  <form id="form1" runat="server">
    <div ng-app="App">
      <div id="div" ng-controller="x">
        <div style="color:green;">
          <a id="up" ng-click="goDown()">Go Down</a></div>
        <div style="color:red;">
          <a id="down" ng-click="goUp()">You're at the bottom</a></div>
      </div>
    </div>
```

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

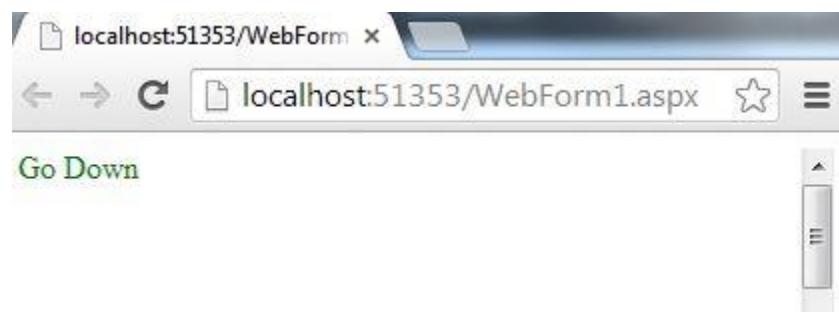
Here a div is bound to the function "x" using the ng-controller, in this div again two div are created. In both the div anchors are used, the first anchor is bound to the "goDown" function using the ng-click, the second anchor is also bound using the ng-click but it is bound to the goUp function.

Some style is also applied to this application, this style will help to show the two anchors at a specific distance by which only one will be seen at a time and the second one will only be seen when the user clicks on the anchor.

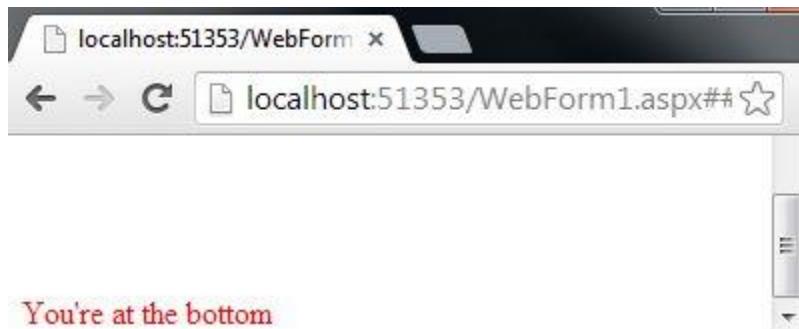
Now our application is created and is ready to for execution.

Output:

Now if we run our application then we will get output like this:



You can see that only one anchor can be seen at this time and in other words the first anchor, now if we click on the first anchor then our page will go down to the element that was called using the JavaScript function.



Complete Code:

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function($scope, $location, $anchorScroll) {
            $scope.goDown = function () {
                $location.hash('down');
                $anchorScroll();
            }
            $scope.goUp = function () {
                $location.hash('up');
                $anchorScroll();
            }
        }
    </script>
    <style>
        #div {
            height: 350px;
            overflow: auto;
        }
        #down {
    
```

```
display: block;
margin-top: 2000px;
}

</style>
</head>
<body>
<form id="form1" runat="server">
<div ng-app="App">
<div id="div" ng-controller="x">
<div style="color:green;">
<a id="up" ng-click="goDown()">Go Down</a></div>
<div style="color:red;">
<a id="down" ng-click="goUp()">You're at the bottom</a></div>
</div>
</div>
</form>
</body>
</html>
```

Logging in Browser Console Using the AngularJS

Introduction

This will explain logging using AngularJS.

Logging means recording the application actions and state in a Secondary Interface, this action can be recorded in an external file, email or anything else, we just need to ensure that it is a secondary interface. For example the Console of our browsers are a secondary interface, so whenever we see output in the console then we are actually logging.

AngularJS provides a service named \$log service, this service is used for logging, we can provide any normal message, any error message, any information message or any warning message using the \$Log Service.

Here I will create an application that will help you understand this service in a convenient way.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official Site. After downloading the external file now you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a JavaScript function in which \$log will be defined along with a default value that will be displayed in the console:

```
<script>
    function x($scope, $log) {
        $scope.$log = $log;
        $scope.consoleMessage = 'Check Your Console';
    }
</script>
```

Here I have created a function named "x", in this function \$scope and \$log are defined. An initial value is passed to a variable named "consoleMessage".

Now our work on the View is completed so we can work on the ViewModel.

Step 3 -

```
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <input type="text" ng-model="consoleMessage"/>
            <input type="button" ng-click="$log.log(consoleMessage)" value="log" />
            <input type="button" ng-click="$log.warn(consoleMessage)" value="Warning" />
            <input type="button" ng-click="$log.info(consoleMessage)" value="Info" />
            <input type="button" ng-click="$log.error(consoleMessage)" value="Error" />
        </div>
    </form>
</body>
```

Here I took a Div bound to the function "x" using ng-controller.

In this Div I took one TextBox and four buttons, the TextBox is bound to the variable consoleMessage using the ng-model so it will be congaing the initial value that was provided in the function.

The first button click is bound to t \$log.log(consoleMessage), in other words a click of this button will show the TextBox message in the console but this message will be a normal message.

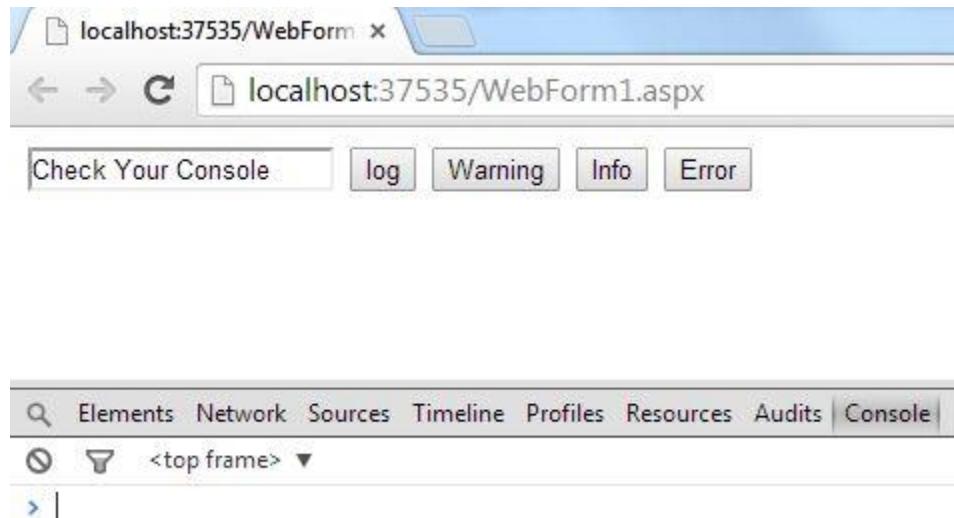
The second button click is bound to \$log.warn(consoleMessage), in other words a click of this button will show the TextBox message in the console but this message will be a warning message with a warning sign.

The third button click is bound to \$log.info(consoleMessage), in other words a click of this button will show the TextBox message in the console but this message will be an info message with an information sign.

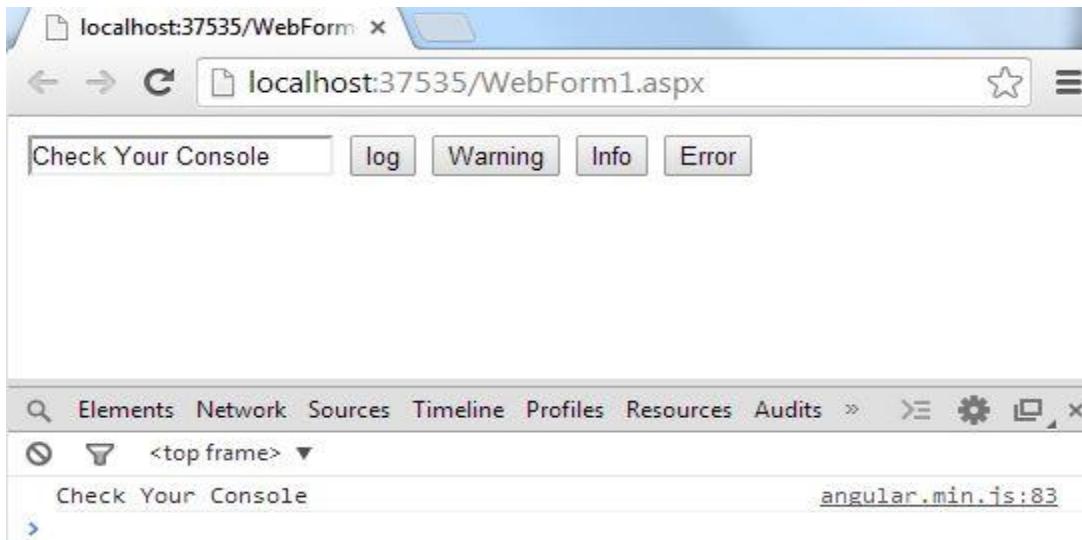
At the end I took a button that is bound to the \$log.error(consoleMessage), in other words a click of this button will show the TextBox message in the console but this message will be an error message with error sign.

Now our application is ready for execution.

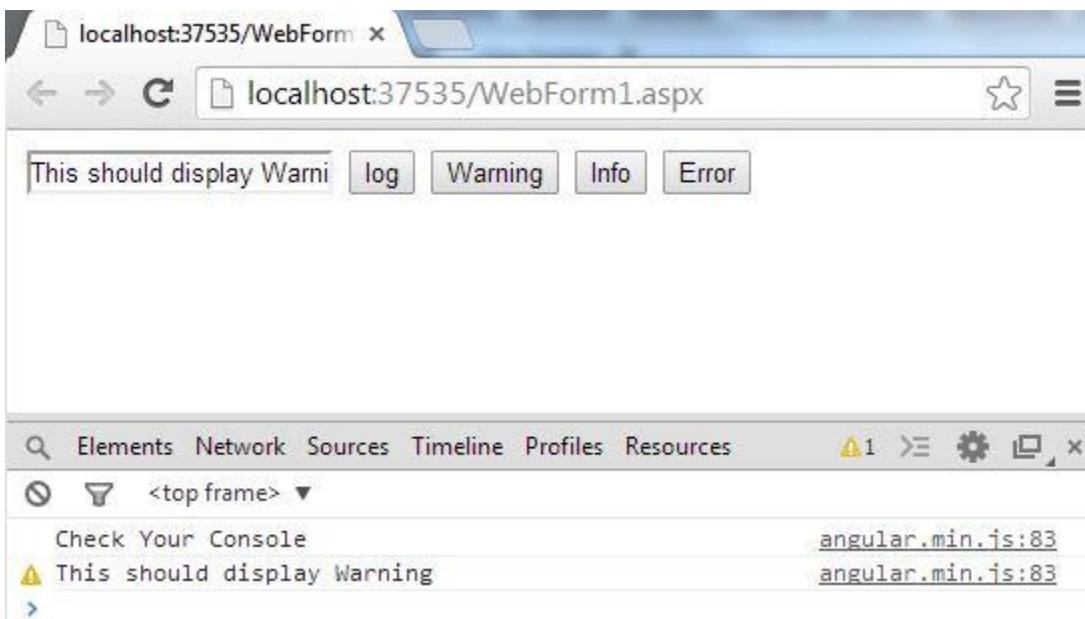
Output: Run your application and for your application output in the Browser you need to open the console in that browser, this can be done either by clicking "F12" or by right-clicking on the browser and then choosing to inspect the element. So your browser will look like this:



You can see that right now my console is clean, the TextBox contains the default value and four buttons are available. Now if I click on the first button then a simple message will be displayed in the console:

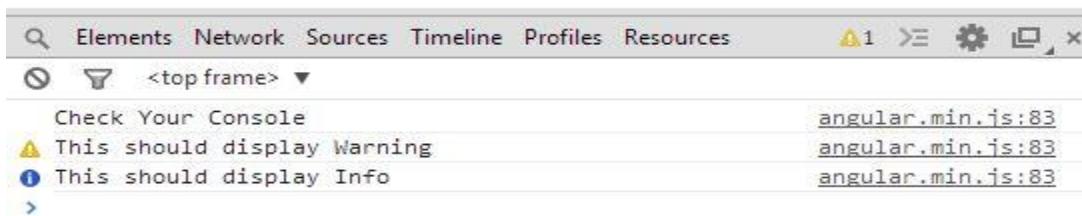
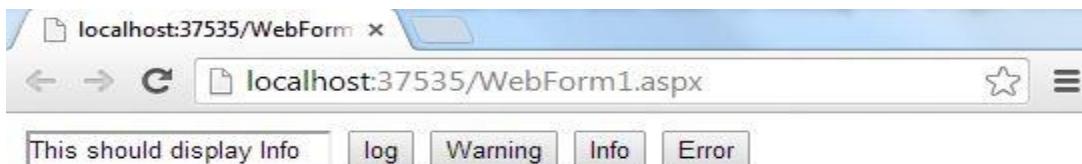


Now I am changing the message in the TextBox and then I will click on the second button that is a warning button:



You can see that I changed the message and the same message is displayed in warning mode.

Now I will again change the text and this time click on the third button, in other words the Info Button:



You can see that the Info message is displayed. Now I will again change the text and this time I will click on the fourth button, in other words the Error Button:



You can see that all types of the messages are shown, also the count of error and warning messages has increased by one in the right hand corner of the console.

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope, $log) {
            $scope.$log = $log;
            $scope.consoleMessage = 'Check Your Console';
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <input type="text" ng-model="consoleMessage"/>
            <input type="button" ng-click="$log.log(consoleMessage)" value="log" />
            <input type="button" ng-click="$log.warn(consoleMessage)" value="Warning" />
            <input type="button" ng-click="$log.info(consoleMessage)" value="Info" />
            <input type="button" ng-click="$log.error(consoleMessage)" value="Error" />
        </div>
    </form>
</body>
</html>
```

\$timeout in AngularJS

Introduction

This explains the \$timeout service provided by AngularJS.

AngularJS provide a service by which you can easily show the time to the external user and you can also use it to provide a condition that will be activated after a certain period of time.

Step 1 - First you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - After adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application where this service will be used and it will be easier for all of you to understand it in this easy way. First I will create a JavaScript function that is the heart of this application:

```
<script>
    function x($scope, $timeout) {
        $scope.miliSeconds = 0;
        var countUp = function () {
            $scope.miliSeconds += 1000;
            $timeout(countUp, 1000); }
        $timeout(countUp, 1000);
    }
</script>
```

Here I have created a function named "x()", in this function \$scope and \$timeout are used, \$timeout is what gets the time and does various other functions.

In the \$scope a variable is defined named miliSeconds, it's value is initially set to 0, so whenever the application is executed the time will be displayed as 0.

Then another function is created named "countUp", in this function the miliSeconds are increased by a thousand and than this value is bound to "countUp" using \$timeout.

Our work on the View is completed, so we can now work on the ViewModel.

Step 3 -

```
<body>
  <form id="form1" runat="server">
    <div ng-controller="x">
      <span>Time (in ms): {{miliSeconds}}</span>
    </div>
  </form>
</body>
```

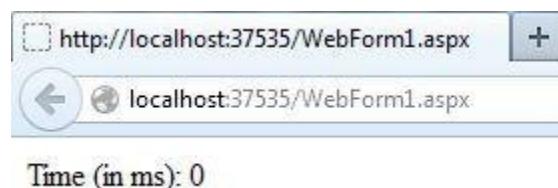
The function name is bound to the div using the ng-controller.

Then miliSeconds is bound to the span, this will update the time after every 1000 miliSeconds.

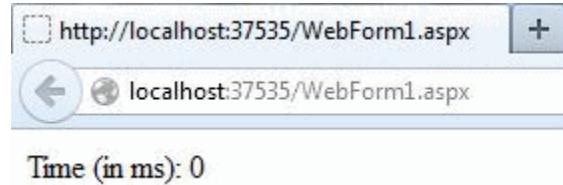
Before all this ensure that <html> is bound using the ng-app, this is necessary since without this binding nothing will be shown in the output.

Now our application is ready for execution.

Output: On running the application your Timer will start from 0.



But then it will be increased by 1000 on every 1000 ms.



The complete code of this application is as follows:

```

<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script>
        function x($scope, $timeout) {
            $scope.miliSeconds = 0;

            var countUp = function () {
                $scope.miliSeconds += 1000;
                $timeout(countUp, 1000);
            }

            $timeout(countUp, 1000);
        }
    </script>
    <script src="angular.min.js"></script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <span>Time (in ms): {{miliSeconds}}</span>
        </div>
    </form>
</body>
</html>

```

\$exceptionHandler in AngularJS

Introduction

In last I explained the **\$timeout in AngularJS**.

This explains the \$exceptionHandler service provided by AngularJS.

AngularJS provide a service by which exceptions can be handled, if any exception occurs then AngularJS provides an error message that will show that an error has occurred.

I will create a simple application that will help you understand this service in a better and easier way.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
  <title></title>
  <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a JavaScript function (that is the important part of this application) as in the following:

```
<script>
  var x = angular.module('except', []);
  x.factory('$exceptionHandler', function () {
```

```

return function (exception, cause) {
    alert(exception.message);
};

});

x.controller('cont', function ($scope) {
    throw { message: 'error occurred' };
});
</script>

```

Here I took a variable "x" in which angular.module is defined. After this factory function is used, the factory function can be used for dependency injection of services but here it will just create an instance of a service whenever it is called. The service used is \$exceptionHandler. In this service we need to provide two things, the first is an exception and the second is a cause, the cause is optional and contains information about the context in which the error was thrown.

After this a controller is created named cont, in this controller the error message is provided.

Now our work on the View is completed so we can work on the ViewModel.

Step 3 -

```

<body>
    <form id="form1" runat="server">
        <div ng-app="except" ng-controller="cont">
        </div>
    </form>
</body>

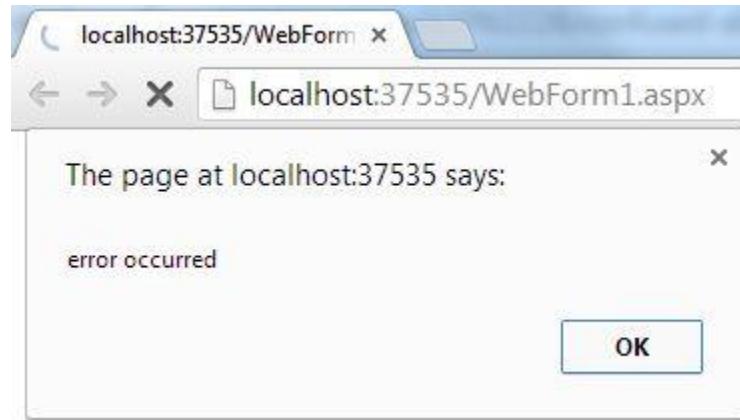
```

Here I have created a simple Div that is bound to the controller and app both.

So whenever we run our application and the div loads then it will show the error message because it is bound to the controller and no condition is provided for an exception to occur.

Now our application is ready to for execution.

Output: On running the application you will see that an error message will be shown when the page was loading:



As you click on "OK" a blank page will be displayed because we have not provided anything in the body section, just a div was created that was also blank.

The complete code of this application is as follows:

```

<html ng-app="except" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        var x = angular.module('except', []);
        x.factory('$exceptionHandler', function () {
            return function (exception, cause) {
                alert(exception.message);  };
        });
        x.controller('cont', function ($scope) {
            throw { message: 'error occurred' };
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-app="except" ng-controller="cont">
        </div>
    </form>
</body>
</html>

```

\$window in AngularJS

Introduction

In last I told you about:

- **\$timeout service provided by the AngularJS.**
- **\$exceptionHandler service provided by the AngularJS.**

This will tell you about the \$window service provided by AngularJS.

In JavaScript "window" is available by default but it can provide many problems because it is a global variable in JavaScript, but in AngularJS we need to declare the \$window service due to which it overrides the already existing "window" and doesn't provide any type of problem while testing the application. If you want to use a window or any property of a window than you first need to declare it.

I will create a simple application that will help you understand this service in a better and easier way.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a JavaScript function that is the important part of this application:

```
<script>
    function x($scope, $window) {
        $scope.greet = 'Hello Anubhav';
        $scope.greetFunc = function (greet) {
            $window.alert(greet);
        };
    }
</script>
```

Here I have created a variable "x", in this function both \$scope and \$window are used, using the scope we will declare the initial values and future functions.

In the variable greet I had passed "Hello Anubhav" as its initial value; that can be changed at run time.

I then created a function named greetFunc, in this function I use the alert property of \$window, this will show the alert message with the text provided in the greet variable.

Now our work on View is completed so we can work on the ViewModel.

Step 3 -

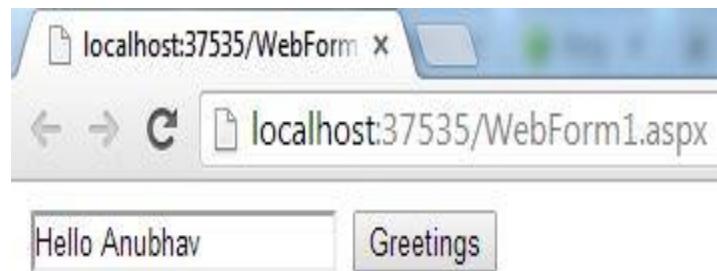
```
<body>
    <form id="form1" runat="server">
        <div ng-app>
            <div ng-controller="x">
                <input type="text" ng-model="greet" />
                <input type="button" ng-click="greetFunc(greet)" value="Greetings"/>
            </div>
        </div>
    </form>
</body>
```

Here I had bound the Div with the function "x", in this div I took a Label and a Button.

Label is bound to the greet using the ng-model. Button is bound to the function "greetFunc" using the ng-click. Now if the user clicks on the button, an alert box will be displayed that will show the default value in the beginning but if the user enters any text in the TextBox then that text will also be displayed.

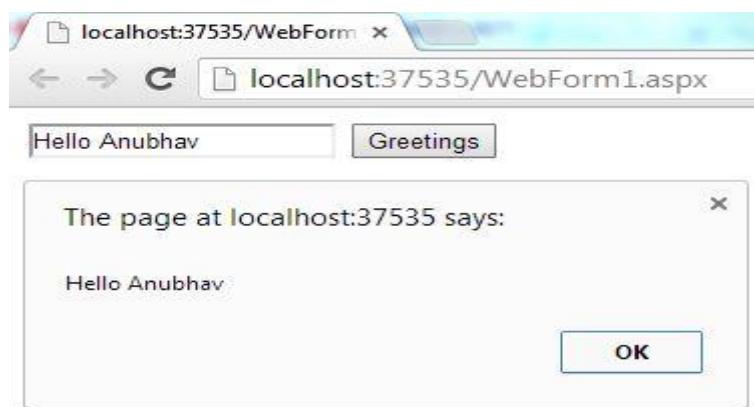
Now our application is ready for execution.

Output: On running the application you will get output like this:

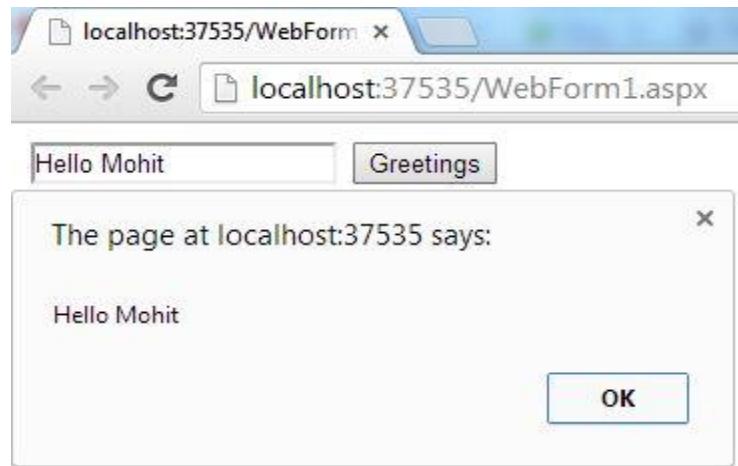


Here you can see that the default value is shown in the TextBox and a button is available.

On the click of a button an alert box will be shown that will display the default Text.



Now if changes are made in the TextBox and again the Button is clicked then you can see that an alert box shows the new value:



The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope, $window) {
            $scope.greet = 'Hello Anubhav';
            $scope.greetFunc = function (greet) {
                $window.alert(greet);
            };
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-app>
            <div ng-controller="x">
                <input type="text" ng-model="greet" />
                <input type="button" ng-click="greetFunc(greet)" value="Greetings"/>
            </div>
        </div>
    </form>
</body>
</html>
```

\$rootScope Service in AngularJS

Introduction

This will tell you about the \$rootScope service provided by AngularJS.

A scope provides a separation between View and its Model. Every application has a \$rootScope provided by AngularJS and every other scope is its child scope.

Here I will create an application that will help you understand this service in a convenient way.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the External JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a JavaScript function in which the \$rootScope service will be initiated.

```
<script>
    angular.module('app', []).controller('x', function ($scope, $rootScope) {
        $rootScope.showAlert = "Hello Everyone";
    });
    angular.module('app').controller('x2', function ($scope, $rootScope) {
        $scope.val = $rootScope.showAlert;
        alert($scope.val);
    });
</script>
```

Here I created two angular.modules, in the first module I created a controller named "x", in this controller the "showAlert" variable is used with the \$rootScope, in this a showAlert message is provided.

In the second controller a variable "val" is used but it is taken under \$scope, the value of rootScope is provided in this variable and then is provided in the alert. Now our work on the View is completed so we can work on the ViewModel.

Step 3 -

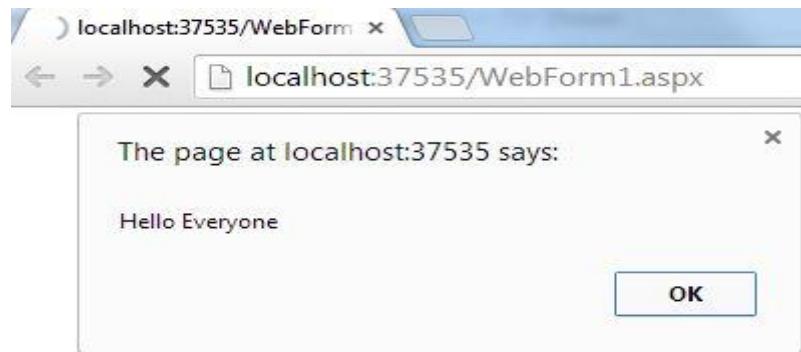
```
<body>
<form id="form1" runat="server">
<div ng-app="app">

<div ng-controller="x"></div>
<div ng-controller="x2">{{val}}</div>
</div>
</form>
</body>
```

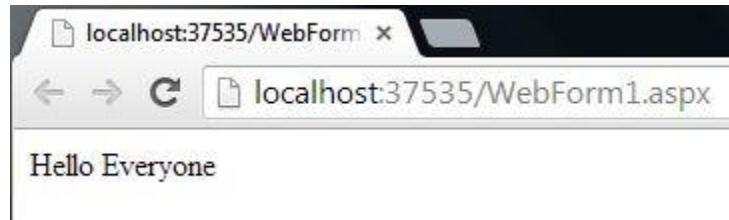
Here I took a Div that is bound using the ng-app, after this two Div are used, one of these is bound to the first controller, "x", and the second is bound to "x2"; both use the ng-controller.

In the second div the "val" variable is bound so this div will display the text that is passed in the Val. Now our application is ready for execution.

Output: On running the application an Alert Message will be displayed while the page is loading:



When you click on the "OK" button the same message will be displayed on the webform as well.



The complete code of this application is as follows:

```
<html ng-app="app" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        angular.module('app', []).controller('x', function ($scope, $rootScope) {
            $rootScope.showAlert = "Hello Everyone";
        });

        angular.module('app').controller('x2', function ($scope, $rootScope) {
            $scope.val = $rootScope.showAlert;
            alert($scope.val);
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-app="app">

            <div ng-controller="x"></div>
            <div ng-controller="x2">{{val}}</div>
        </div>
    </form>
</body>
</html>
```

Enter and Leave Animation Using AngularJS

Introduction

This explains how to enter and leave Animation in AngularJS.

In this I use AngularJS 1.2.2, if you are using 1.2.11 then you need to first add 1.2.2 and remove 1.2.11 from your application otherwise Animation will not be supported.

CSS Animation will be used to help Angular Application Animate.

The ng-animate directive is not supported in AngularJS 1.2 Version, instead we need to Add "Angular.Module" and then "ngAnimate" will be added in it.

Step 1 - First you need to pass the reference of two external Angular files or you can also download them and then can add them directly to the head section, these files are:

- angular.js 1.2.2
- angular-animate.min.js

You can pass the reference of these files in this format:

```
<head runat="server">
    <title></title>
    <script src="http://code.angularjs.org/1.2.2/angular.js"></script>
    <script src="http://code.angularjs.org/1.2.2/angular-animate.min.js"></script>
</head>
```

Step 2 - After adding these external files you need to work on the JavaScript of this application, add this code to the head section:

```
<script>
    angular.module('animate', ['ngAnimate'])
        .controller('animateCtrl', function (
            $scope
        ) {
            $scope.names = [{
```

```

        name: 'Mohit'
    }, {
        name: 'Anubhav' }];
$scope.insert = function () {
    $scope.names.push({
        name: 'Anubhav'
    });
    $scope.remove = function () {
        $scope.names.pop();
    };
});
</script>

```

As I said earlier if you want to animate then you need to add the angular.module, I had first added it, this angular.module contains a variable named "animate". This is the variable that will bind using the ng-app, after this I had passed the ngAnimate.

The controller name is "animateCtrl". In this controller function I had passed some initial values using the \$scope.

After this a function is created named "insert" function, this function will help to insert a new value in the already existing data on the click of a button, for this to happen I have used push().

One more function is created that is named "remove" function, this function will help to remove the data on the click of a button, for this to happen I have used pop().

Step 3 - Until now our work on the View is completed and now we need to work on the View Model of this application.

Write this code in the body section of your application:

```

<body ng-controller="animateCtrl">
<div>
    <button ng-click="insert()">Enter Animation</button>
    <button ng-click="remove()">Leave Animation</button>
    <ul>
        <li ng-repeat="user in names">
            <a>{{user.name}}</a>
        </li>
    </ul>
</div>

```

```
</ul>
</div>
</body>
```

Here I had bound the body with the ng-controller. I used a Div in which two buttons are used.

You can see that the first button click is bound to the insert() function using ng-click and the second button is bound to the remove() funciton using ng-click.

After the buttons I created a dynamic list, this list is dynamic because I am using the ng-repeat directive in this list, this will repeat the functionality for each new data so a new row will be added for each new data.

Step 4 - Now we need to add some CSS to this application.

```
<style>
.ng-enter
{
    -webkit-transition: 1s;
    transition: 1s;
    margin-left: 100%;
}
.ng-enter-active
{
    margin-left: 0;
}

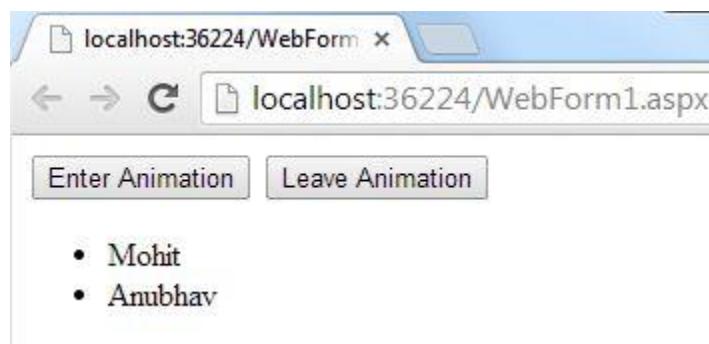
.ng-leave
{
    -webkit-transition: 1s;
    transition: 1s;
    margin-left: 0;
}
.ng-leave-active
{
    margin-left: 100%;
}
</style>
```

As the Angular.Module was necessary in the JavaScript Section similarly "-webkit-transition" is necessary in the CSS section, if transition is removed then our Animation will stop working.

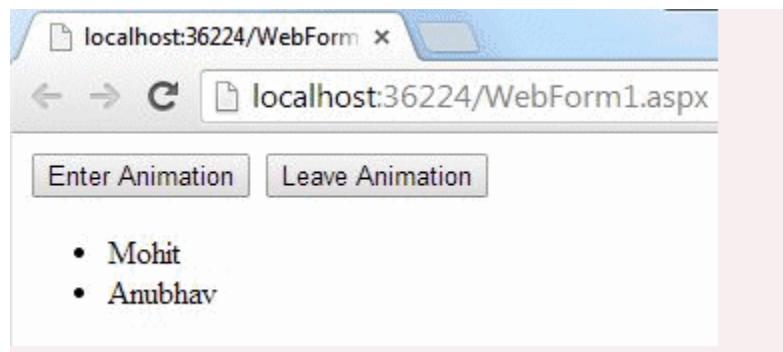
Here I have created two classes for each enter and leave, the second class of both enter and leave will work when they are in active mode.

Now our application is completed and is ready for execution.

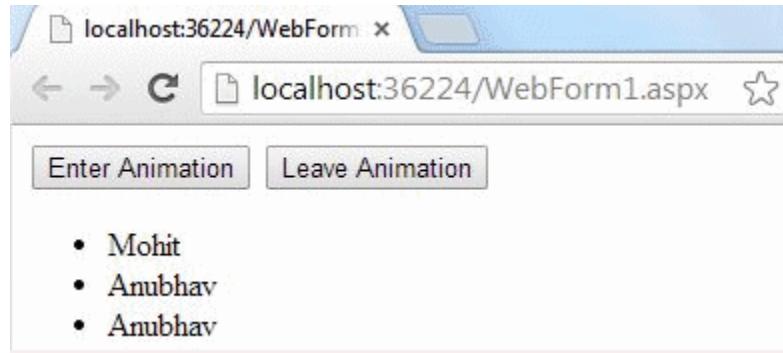
Output: On running the application you will get output like this:



First only two values will be shown, but as you click on "Enter Animation", an animation like this will be shown:



But as you will click on the leave button, the data will leave as in this animation:



The complete code of this application is as follows:

```
<html ng-app="animate" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="http://code.angularjs.org/1.2.2/angular.js"></script>
    <script src="http://code.angularjs.org/1.2.2/angular-animate.min.js"></script>
    <style>
        .ng-enter
        {
            -webkit-transition: 1s;
            transition: 1s;
            margin-left: 100%;
        }
        .ng-enter-active
        {
            margin-left: 0;
        }
        .ng-leave
        {
            -webkit-transition: 1s;
            transition: 1s;
            margin-left: 0;
        }
        .ng-leave-active
        {
            margin-left: 100%;
        }
    </style>
</head>
<body>
    <div ng-controller="ctrl">
        <button ng-click="addName('Mohit')">Enter Animation</button>
        <button ng-click="removeName('Mohit')">Leave Animation</button>
        <ul>
            <li>• Mohit</li>
            <li>• Anubhav</li>
            <li>• Anubhav</li>
        </ul>
    </div>
</body>
</html>
```

```
</style>

<script>
    angular.module('animate', ['ngAnimate'])
        .controller('animateCtrl', function (
            $scope
        ) {
            $scope.names = [
                {
                    name: 'Mohit'
                },
                {
                    name: 'Anubhav'
                }];
            $scope.insert = function () {
                $scope.names.push({
                    name: 'Anubhav'
                });
            };
            $scope.remove = function () {
                $scope.names.pop();
            };
        });
</script>
</head>

<body ng-controller="animateCtrl">
    <div>
        <button ng-click="insert()">Enter Animation</button>
        <button ng-click="remove()">Leave Animation</button>
        <ul>
            <li ng-repeat="user in names">
                <a>{{user.name}}</a>
            </li>
        </ul>
    </div>
</body>
</html>
```

Add Animation To Your Application Using AngularJS

Introduction

This explains how to add Animation to your application using AngularJS.

Angular 1.1.5 provides a directive known as ng-animate, this directive is used to add animation to a application, but Angular 1.2 has changed everything and doesn't support ng-animate.

This article uses Angular 1.1.5, the previous version of Angular, so if you are adding Angular 1.2 then you need to remove this JS file and need to add 1.1.5 or it's reference to your application.

Step 1 - First you need to add angular 1.1.5, this can be either downloaded from the Angular official website or you can add this reference to the head section of your application:

```
<head runat="server">
    <title></title>
    <script src="http://code.angularjs.org/1.1.5/angular.min.js"></script>
</head>
```

Step 2 - Now I will work on the View of this application, in other words on the JavaScript part of this application.

```
<script>
    angular.module('animate', [
        ]).controller('animateCtrl', function (
            $scope ) {
        $scope.names = [
            { name: 'Mohit' },
            { name: 'Anubhav' }];
        $scope.insert = function () {
            $scope.names.push({
                name: 'Anubhav'
            });
        });
    </script>
```

Here I have created an Angular.module named "animate", in this module I have created a controller named "animateCtrl".

In this Controller function I have provided some Initial name.

After this I had created a function that will insert a new name in the list of names.

Now our work on View is completed and we can move toward the View Model in other words on the HTML part.

Step 3 -

```
<body ng-controller="animateCtrl">
<div>
  <button ng-click="insert()">Add More Anubhav</button>
  <ul>
    <li ng-repeat="user in names" ng-animate="{enter: 'animated flip'}">
      <a>{{user.name}}</a>
    </li>
  </ul>
</div>
</body>
```

Here I had bound the body section with the controller "animateCtrl" using the ng-controller.

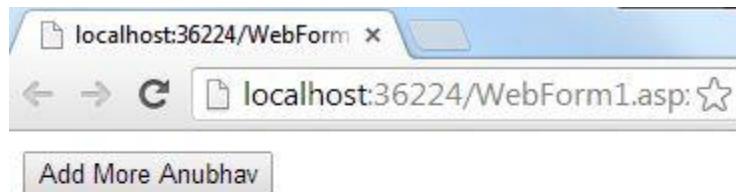
After this I took a button, the click of this button is bound with the insert() function using ng-click.

At the end I created a List bound with the names passed in the controller, this is a dynamic list created by ng-repeat. User. name will help to show the data in each new row of the list.

In the ng-animate I passed "enter:'animated flip'", this is a type of Animation that will be shown when amaking a new entry using the click of a button.

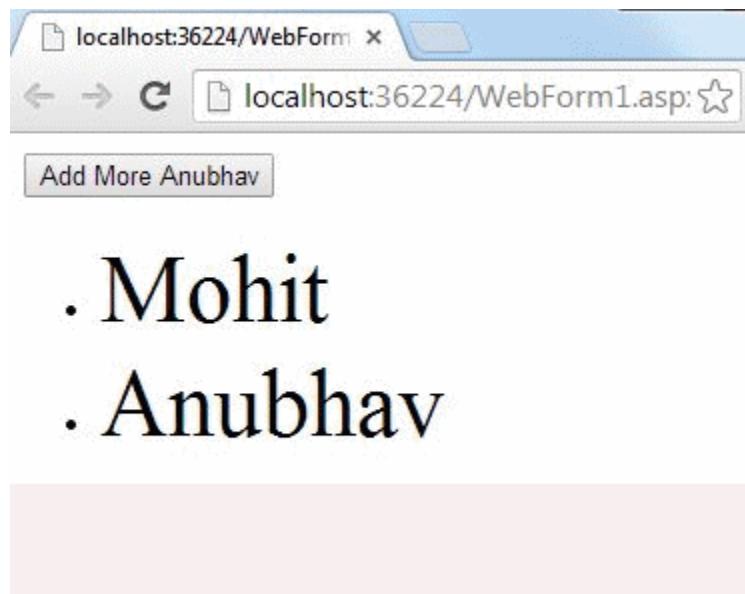
Now our application is created and is ready for execution.

Output: On running the application you will get an output like this one:



. Mohit
. Anubhav

Now as I click on the Button then a new name will be added with this Animation:



The complete code of this application is as follows:

```
<html ng-app="animate" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link rel="stylesheet" href="https://rawgithub.com/daneden/animate.css/master/animate.css">
```

```
<style>
a {
  font-size: 300%;
}
</style>

<script src="http://code.angularjs.org/1.1.5/angular.min.js"></script>
<script>
angular.module('animate', [
]).controller('animateCtrl', function (
  $scope
) {
  $scope.names = [
    {
      name: 'Mohit'
    },
    {
      name: 'Anubhav'
    }];
  $scope.insert = function () {
    $scope.names.push({
      name: 'Anubhav'
    });
  };
});
</script>
</head>

<body ng-controller="animateCtrl">
<div>
  <button ng-click="insert()">Add More Anubhav</button>
  <ul>
    <li ng-repeat="user in names" ng-animate="{enter: 'animated flip'}">
      <a>{{user.name}}</a>
    </li>
  </ul>
</div>
</body>
</html>
```

\$rootElement Service in AngularJS

Introduction

This explains the \$rootElement Service provided by AngularJS.

AngularJS provides a service named \$rootElement, by using this service the user can display the name of the element where "ng-app" is declared or where the element was passed into the angular.Bootstrap. This service can also be used to learn the location of where the injector is published, knowing this type of location we need to use \$rootElement.injector().

Here I will create an application that will help you understand this service in a convenient way.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a JavaScript function in which the \$rootElement Service will be initiated.

```
<script>
    var showAppName = angular.module('showAppName', []);
    showAppName.service('serve', function ($rootElement) {
        return { appName: $rootElement.attr('ng-app') };
    });
    showAppName.controller('x', function ($scope, serve) {
```

```

$scope.appName = serve.appName;
});
</script>

```

Here I took a variable "showAppName", in this variable I took the angular.module in which the value for ng-app is provided, this is the name that will be bound to the ng-app and will be displayed in the output window.

Then I created a function in which the \$rootElement Service is declared, since we want to display only the ng-app value so I am using the \$rootElement.attr, if we want to display the location of where the injector is published then we need to declare the \$rootElement.injector().

After this I have created a controller function named "x", the value of ng-app is passed to a variable named as appName.

Now our work on the View is completed so we can work on the ViewModel.

Step 3 -

```

<body>
  <form id="form1" runat="server">
    <div ng-app="showAppName" ng-controller="x">
      This is my App Name: "{{appName}}"
    </div>
  </form>
</body>

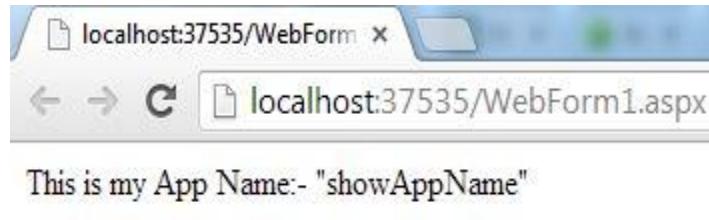
```

Here I took a div that is bound to the ng-app and ng-controller both, in the ng-app I passed the "showAppName" that was declared in the first line of JavaScript code, in the Controller I have provided the name of the controller function.

After this I had written some text that can also be provided inside the label, appName is bound with this text using the {{ }} brackets, this will show the value that was provided in the first line of code.

Now our application is ready for execution.

Output: On running the application you will see that the ng-app value is provided along with the text that was provided in the DOM.

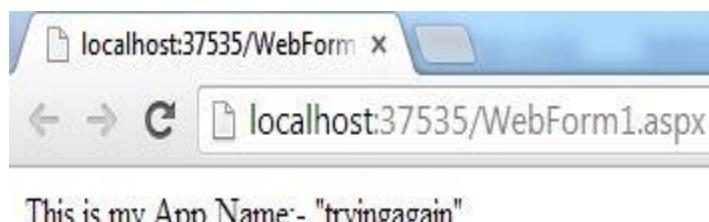


You can see that the expected result is shown in the output window.

Now I am changing the ng-app value in the code:

```
<script>
  var tryingagain = angular.module('tryingagain', []);
  tryingagain.service('serve', function ($rootElement) {
    return { appName: $rootElement.attr('ng-app') };
  });
  tryingagain.controller('x', function ($scope, serve) {
    $scope.appName = serve.appName;
  });
</script>
```

I am running the code again and a different result can be seen:



The complete code of this application is as follows:

```
<html ng-app="tryingagain" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        var tryingagain = angular.module('tryingagain', []);
        tryingagain.service('serve', function ($rootScope) {
            return { appName: $rootScope.$attr['ng-app'] };
        });

        tryingagain.controller('x', function ($scope, serve) {
            $scope.appName = serve.appName;
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-app="tryingagain" ng-controller="x">
            This is my App Name:- "{{appName}}"
        </div>
    </form>
</body>
</html>
```

Fade In and Fade Out Animation Using AngularJS

Introduction

This explains how to add Fade In and Fade Out Animation to your application using AngularJS.

In this I will use AngularJS 1.2.2, if you are using 1.2.11 then you need to first add 1.2.2 and remove 1.2.11 from your application otherwise Animation will not be supported.

CSS Animation will be used to help Angular Application Animate.

The ng-animate directive is not supported in AngularJS 1.2 Version, instead we need to add "Angular.Module" and then "ngAnimate" will be added to it.

Step 1 - First you need to pass the reference of two external Angular files or you can also download them and then add them directly in the head section, these files are:

- angular.js 1.2.2
- angular-animate.min.js

You can pass the reference of these files in this format:

```
<head runat="server">
    <title></title>
    <script src="http://code.angularjs.org/1.2.2/angular.js"></script>
    <script src="http://code.angularjs.org/1.2.2/angular-animate.min.js"></script>
</head>
```

Step 2 - After adding these external files you need to work on the JavaScript of this application, add this code in the head section:

```
<script>
    angular.module('animate', ['ngAnimate'])
        .controller('animateCtrl', function (
            $scope
        ) {
            $scope.names = [{
```

```

        name: 'Mohit'
    }, {
        name: 'Anubhav'
    }];
$scope.insert = function () {
    $scope.names.push({
        name: 'Anubhav'
    });
};
$scope.remove = function () {
    $scope.names.pop();
};
});
</script>

```

As I told you earlier, if you want to animate then you need to add the angular.module, I first added it. This angular.module contains a variable named "animate"; this is the variable that will be bound using the ng-app, after this I passed the ngAnimate.

"animateCtrl" is the controller name. In this controller function I passed some initial values using \$scope.

After this a function is created named insert, this function will help to insert a new value into the previously existing data on the click of the button, for this to happen I used push().

One more function is created named remove, this function will help to remove the data on the click of a button, for this to happen I used pop().

Step 3 - Until now our work on the View is completed and now we need to work on the View Model of this application.

Write this code in the body section of your application:

```

<body ng-controller="animateCtrl">
<div>
    <button ng-click="insert()">Enter Animation</button>
    <button ng-click="remove()">Leave Animation</button>
    <ul>
        <li ng-repeat="user in names">

```

```

<a>{{user.name}}</a>
</li>
</ul>
</div>
</body>

```

Here I had bound the body with the ng-controller, I took a Div in which two buttons are used.

You can see that the first button click is bound to the insert() function using ng-click and the second button is bound to the removed() function using ng-click.

After the buttons I created a dynamic list, this list is dynamic because I am using the ng-repeat directive in this list, this will repeat the functionality for each new data so a new row will be added for each new data.

Step 4 - Now we need to add some CSS to this application.

```

<style>
.ng-enter
{
    -webkit-transition: 1s;
    transition: 3s;
    opacity:0;
}
.ng-enter-active
{
    opacity:1;
}
.ng-leave
{
    -webkit-transition: 1s;
    transition: 2s;
}
.ng-leave-active
{
    opacity:0;
}
</style>

```

As the Angular.Module was necessary in the JavaScript Section similarly "-webkit-transition" is necessary in the CSS section, if transition is removed then our animation will stop working.

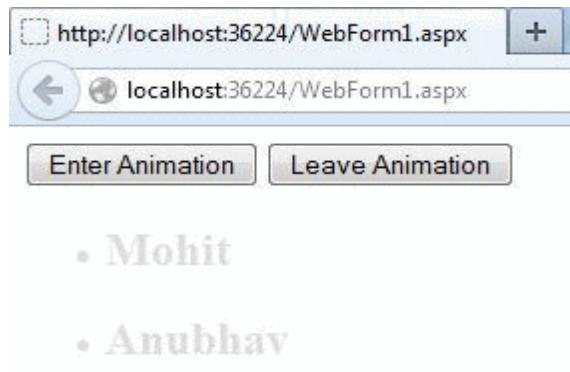
Here I created two classes for each enter and leave, the second class of both the enter and leave will work when they are in the active mode.

In the ng-enter class the opacity is set to 0 which means that initially the data will not be seen, since the user will click on the first button the ng-enter-active class will be activated and its opacity will be changed to 1, in other words that data will start showing but with a delay of 3s.

Similarly in the ng-leave-active class the opacity is set to 0, in other words the click on the second button will hide the data but with a delay of 2s.

Now our application is completed and is ready for execution.

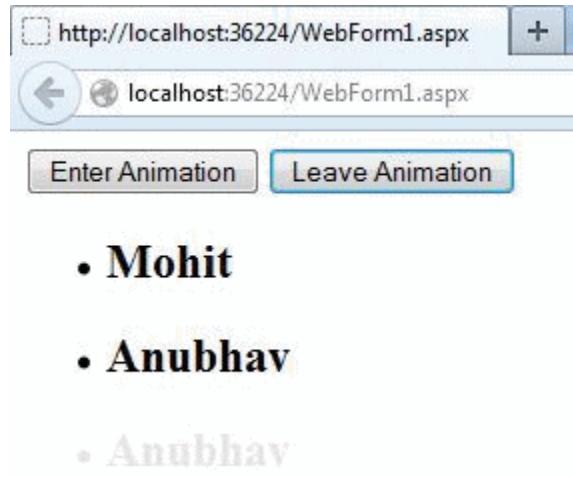
Output: On running the application you will get an output like this one:



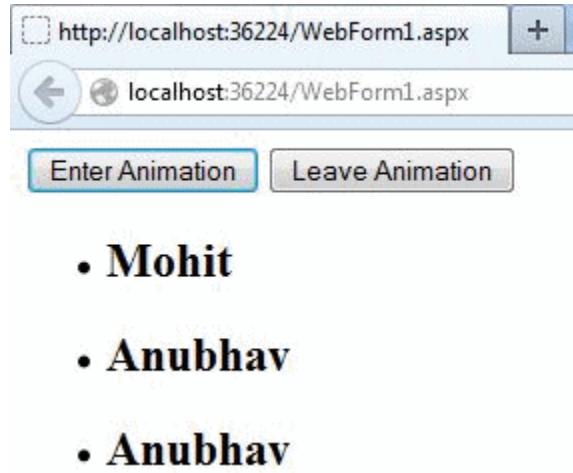
You might be thinking that the opacity was 0 by default, so why are these two values appearing on running the application?

That's because these two values were the default value that were passed in the JavaScript function.

Now if I click on the Enter Animation then a new entry will be made and that's too with a animation affect.



Similarly if I click on the second button then last entry will be removed with an animation affect.



Use ng-class With CSS3 Transition in AngularJS

Introduction

This will tell you how to use ng-class with CSS3 Transition in AngularJS.

In previous I have also used CSS but in there the classes were made that could take place in any version of CSS; in this I will use CSS3 with the ng-class in AngularJS.

Step 1 - First you need to provide the reference to AngularJS file in the head section, you can also download it and can apply it directly in your application.

```
<head runat="server">
    <title></title>
    <script src="http://code.angularjs.org/1.0.7/angular.min.js"></script>
</head>
```

Step 2 - After providing the reference you need to work on the JavaScript of the application.

Write this code in the head section of your application:

```
<script>
    angular.module('mod', [ ]).controller('x', function (
        $scope
    ) {
        $scope.blur = false;
    });
</script>
```

Here I had simply created an "angular.module" and named it "mod", in this module a controller function is created named "x()".

In this function I initialized the \$scope and with it a variable is used named blur, by default it is made false.

Step 3 - Now I will work on the View Model of this application.

```
<body ng-controller="x">
  <button ng-click="blur = !blur">Toggle Fade</button>
  <h1 ng-class="{ blur: blur }">Hey Anubhav!!</h1>
</body>
```

Here the body is bound with the function "x()" using the ng-controller.

In this body I have created a button that is bound with blur using the ng-click.

Also a <h1> tag is used that is bound with blur, whatever the data provided in this blur will have the blur class that will be created in our next step.

Step 4 - Now I will work on the CSS of this application and will create the blur class.

```
<style>
  h1 {
    -webkit-transition: 1s;
    transition: 1s;
    opacity: 1;
  }
  h1.blur {
    -webkit-transition: 1s;
    transition: 1s;
    opacity: 0.2;
  }
</style>
```

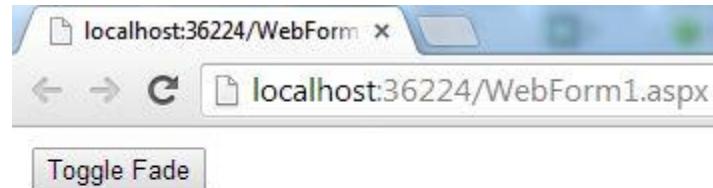
You can see that I have created two classes first for the <h1> and second for the blur in <h1>.

As you already know that for applying the CSS to the AngularJS application you need to apply the "-webkit-transition", so I had applied it in both the classes.

h1 opacity is set to 1, in other words no opacity but in "h1.blur" the opacity is set to 0.2, in other words the data will be blurred on the click of a button.

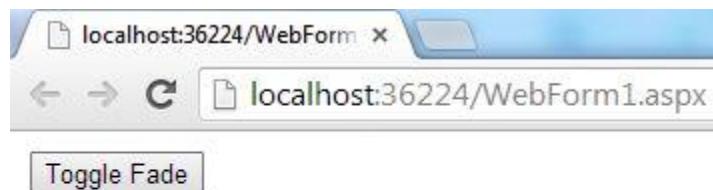
Now our application is created and can be executed.

Output: On running the application you will see the default text in h1 format:



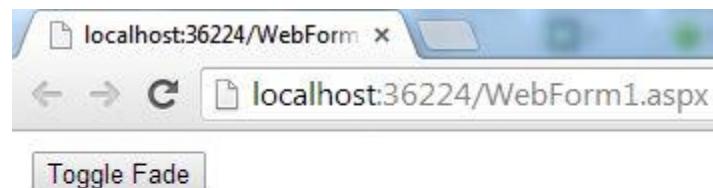
Hey Anubhav!!

Now if I click on the button then the data will become faded:



Hey Anubhav!!

Now if I again click on the button then again the text will be shown clearly.



Hey Anubhav!!

The complete code of this application is as follows:

```
<html ng-app="mod" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="http://code.angularjs.org/1.0.7/angular.min.js"></script>
    <style>
        h1 {
            -webkit-transition: 1s;
            transition: 1s;
            opacity: 1;
        }

        h1.blur {
            -webkit-transition: 1s;
            transition: 1s;
            opacity: 0.2;
        }
    </style>

    <script>
        angular.module('mod', [ ]).controller('x', function (
            $scope
        ) {
            $scope.blur = false;
        });
    </script>
</head>

<body ng-controller="x">
    <button ng-click="blur = !blur">Toggle Fade</button>
    <h1 ng-class="{ blur: blur }">Hey Anubhav!!</h1>
</body>
</html>
```

Include Multiple Partial Templates in Single Application Using Ng-Template and Config

Introduction

This explains how to include multiple Partial Templates in a single application using ng-Template and config.

Various kinds of text can be shown in different templates using AngularJS. For this we need to use various things like ng-Template and config, config is used just like a controller and ng-template is defined in the script tag.

Let's create a sample application that will help you understand this feature more interestingly.

Step 1 - First of all we will work on the JavaScript of this application but before that you need to add an external link to your application as in the following:

```
<head runat="server">
    <title></title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.5/angular.js"></script>
</head>
```

After adding this external link we can work on the JavaScript of this application.

Step 2 - Add this code to the head section of your application:

```
<script>
    var x = angular.module('app', []);
    x.config(function ($routeProvider) {
        $routeProvider.when('/first', { templateUrl: 'firstPage.html' })
            .when('/second', { templateUrl: 'SecondPage.html' });
    });

    x.controller('control', function ($scope) {
        $scope.test = 'Hello';
   });  </script>
```

Here first I have created an angular.module named "app", this will be bound with the ng-app.

After this config is called using the variable x that was declared in the angular.module.

In this config "\$routeProvider" is used, routeProvider is used to set the path of the template that is to be bound with the application. This routeProvider will check whether the user has clicked on the "first" or the "second", if the first link is clicked then it will pass the URL of the template that is bound on this click and if the second link is clicked then the second path will be passed.

After this controller is created named "control", in this controller a variable is used in which a default value is passed.

Step 3 - Now I will create a template, for this you need to write this code in the body section:

```
<script type="text/ng-template" id="firstPage.html">  
    This is in the First Page  
</script>
```

As I have already said, to use the ng-template you need to use the script tag, here you can also see that I used the <script> tag in which the type is provided as ng-template and in this script some text is provided that will be shown while running the application.

Similarly I have created the second template, so the body section looks like this:

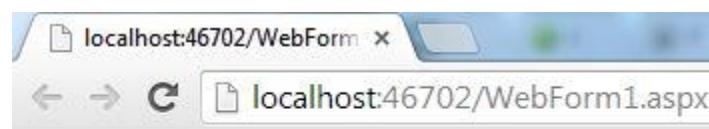
```
<body ng-app="app" ng-controller="control">  
    <script type="text/ng-template" id="firstPage.html">  
        This is in the First Page  
    </script>  
  
    <script type="text/ng-template" id="SecondPage.html">  
        This is in the Second Page  
    </script>  
    <ul>  
        <li><a href="#/first">First Page</a></li>  
        <li><a href="#/second">Second Page</a></li>  
    </ul>  
  
<div ng-view>  
    Sry, Some Error Occured </div> </body>
```

After creating the two templates I used the List in which two Anchors are used, in the first anchor "#/first" is passed in the href and in the second anchor "#/second" is passed in the href, in other words that first link is bound with the first template and the second link is bound with the second template.

At the end a div is taken that is bound with the ng-view so this div that will show the text on the click of Anchors.

Now our application is created and is ready for execution.

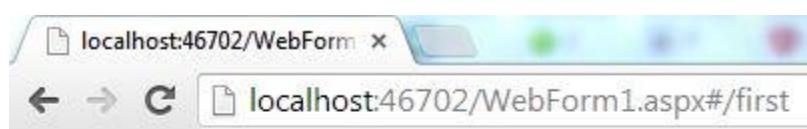
Output: On running the application you will get an output like this one:



- [First Page](#)
- [Second Page](#)

Here you can see that two Anchors are available but no text is shown; that's because no Anchor is clicked.

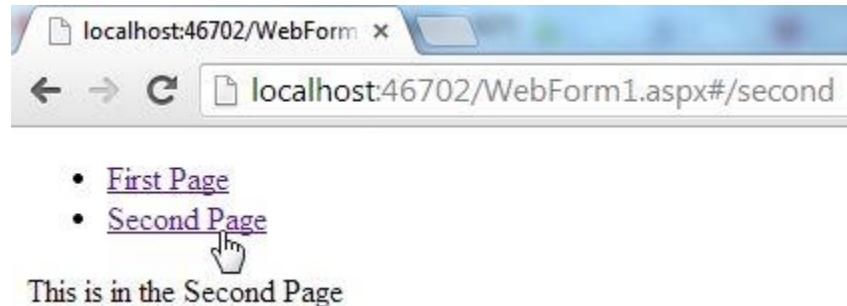
But as I click on the first Anchor then the text of the first Template will be shown as in the following:



- [First Page](#)
- [Second Page](#)

This is in the First Page

Click on the second Anchor that will show the data of the second Template as in the following:



The complete code of this application is as follows:

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.5/angular.js"></script>
    <script>
        var x = angular.module('app', []);
        x.config(function ($routeProvider) {
            $routeProvider.when('/first', { templateUrl: 'firstPaget.html' })
                .when('/second', { templateUrl: 'SecondPage.html' });
        });

        x.controller('control', function ($scope) {
            $scope.test = 'Hello';
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <body ng-app="app" ng-controller="control">
            <script type="text/ng-template" id="firstPaget.html">
                This is in the First Page
            </script>

            <script type="text/ng-template" id="SecondPage.html">
                This is in the Second Page
            </script>
        <ul>
            <li><a href="#/first">First Page</a></li>

```

```
<li><a href="#/second">Second Page</a></li>
</ul>

<div ng-view>
    Sry, Some Error Occured
</div>
</body>
</form>
</body>
</html>
```

Compare Two Values Using angular.equals

Introduction

In this I will tell you how to compare two values using angular.equals.

We often create applications where we need to compare a few things just as in the case of entering a password and re-enter the password. In such a cases if the user is not providing the equal values then an error message needs to be shown. AngularJS provides a feature for doing that easily, angular.equals.

As we already know, using AngularJS we can create dynamic things that can be changed according to the user requirements, so here I will create an application that can compare dynamic data.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the External JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the View/JavaScript part of this application. Write this code in the head section:

```
<script>
    function x($scope) {
        $scope.firstPassword = 'Anubhav';
```

```

$scope.secondPassword = 'Chaudhary';
$scope.func = function () {
    alert(angular.equals($scope.firstPassword, $scope.secondPassword));
}
}
</script>

```

I have created a function and named it "x", in this function a few initial values are provided.

These initial values will be provided in the textboxes where the password needs to be entered and reentered.

After this I have created a function "func()", this function will compare the values of these variables and will show an alert message accordingly.

You can see that for comparing the values I have used angular.equals, and in the bracket I provided both of the variables along with \$scope. Now our View or JavaScript part is completed and we can work on the ViewModel of our application.

Step 3 - Now I will work on the design or ViewModel part, write this code in the body section:

```

<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <label>First Password: </label><input type="password" ng-model="firstPassword" />
            <br />
            <label>Re Enter Password: </label><input type="password" ng-model="secondPassword" />
            <br />
            <br />
            <label ng-bind-template="Hello! {{firstPassword}} {{secondPassword}}"></label>
            <input type="button" ng-click="func()" value="Test"/>
        </div>
    </form>
</body>

```

Here I took two Textboxes whose type is set to "password".

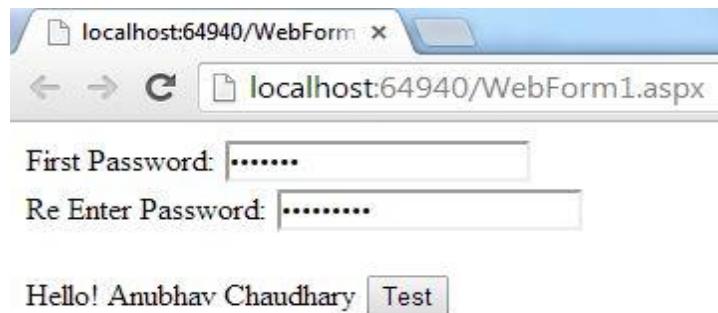
The first TextBox is bound to "firstPassword" and the second is bound to "secondPassword". This binding will provide the default values in these Textboxes.

After this a label is taken in which both the default passwords are bounded, this can be done using the ng-bind-template.

At the end a button is taken, on the click of this button the function func() is called.

Now our application is created and is ready for execution.

Output: On running the application you will see that some text is available in both of the textboxes in password format. Whatever is written in the Textboxes can be seen in the Label present after them.

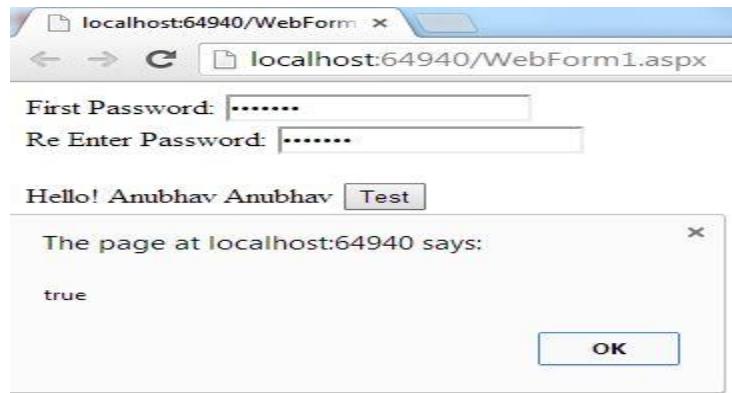


The screenshot shows a web browser window with the URL `localhost:64940/WebForm1.aspx`. It contains two text input fields labeled "First Password" and "Re Enter Password", both containing the value ".....". Below these fields is a label "Hello! Anubhav Chaudhary" followed by a button labeled "Test".

If I click on the button then a message will be shown that will show "false"; that's because the values were different:



Now I am changing the text in the second TextBox and making it similar to the first one (you can check the Label). After making the changes I clicked on the button and a True message is displayed as in the following:



The complete code of this application is as follows:

```

<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.firstPassword = 'Anubhav';
            $scope.secondPassword = 'Chaudhary';
            $scope.func = function () {
                alert(angular.equals($scope.firstPassword, $scope.secondPassword)); } }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <label>First Password: </label><input type="password" ng-model="firstPassword" />
            <br />
            <label>Re Enter Password: </label><input type="password" ng-model="secondPassword" />
            <br />
            <br />
            <label ng-bind-template="Hello! {{firstPassword}} {{secondPassword}}"></label>
            <input type="button" ng-click="func()" value="Test"/>
        </div>
    </form>
</body>
</html>

```

Ng-Include Directive of AngularJS

Introduction

In previous I told you about:

- **ng-click and ng-blur Directive of AngularJS**
- **ng-init and ng-repeat Directive of AngularJS**
- **ng-change Directive of AngularJS**
- **ng-BindHtml Directive of AngularJS**
- **ng-bind-template directive of AngularJS.**
- **ngClassOdd and ngClassEven Directive of AngularJS**
- **ng-cut, ng-copy and ng-paste Directive of AngularJS**
- **ng-disabled Directive of AngularJS**
- **ng-show Directive of AngularJS.**
- **ng-hide Directive of AngularJS.**
- **ng-dblclick Directive of AngularJS**
- **ng-keyup and ng-keydown Directive of AngularJS**
- **ng-Switch Directive of AngularJS**
- **ng-Transclude Directive of AngularJS**
- **ng-Value Directive of AngularJS**
- **How to Include Multiple Partial Template in Single Application Using ng-Template and Config.**

This explains the ng-Include Directive of AngularJS.

In previous I showed how multiple partial templates can be included in a single application, ng-include is another method that can help us to include multiple templates.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
<title></title>
<script src="angular.min.js">
</script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application that will help you understand this directive.

First I will create a JavaScript function in which some initial values will be passed.

```
<script>
function x($scope) {
    $scope.templatePages =
        [{ temp: 'firstPage.html', url: 'firstPage.html' }
        , { temp: 'secondPage.html', url: 'secondPage.html' }];
    $scope.tempPage = $scope.templatePages[0];
}
</script>
```

Here I first created a function named "x", in this function a variable is declared, in this variable the name of our templates and their URL are linked.

Step 3 -Now I will work on the View Model of this application.

```
<div ng-app>
<div ng-controller="x">
    <select ng-model="tempPage" ng-options="p.temp for p in templatePages">
        <option value="">(Empty)</option>
    </select>

    <div ng-include="tempPage.url">
    </div>
</div>
```

```
<script type="text/ng-template" id="firstPage.html">
    Hey!! You are on First template
</script>
<script type="text/ng-template" id="secondPage.html">
    Hey!! You are on Second template
</script>
</div>
```

The function name is bound to the Div using the ng-controller.

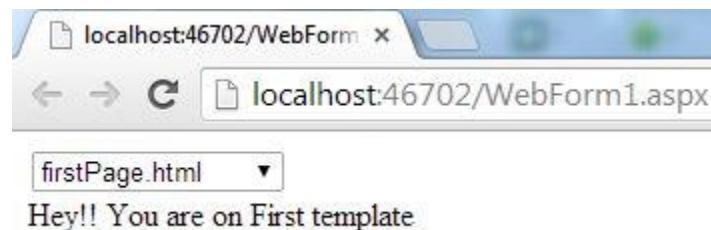
After this a dynamic list is created using the ng-options, this list will be shown in the drop down list.

A dynamic list will contain the URL of templates that are included in our application. Whatever name or URL is selected from the drop down regarding the data will be shown in the div that is bound to tempPage using the ng-include directive.

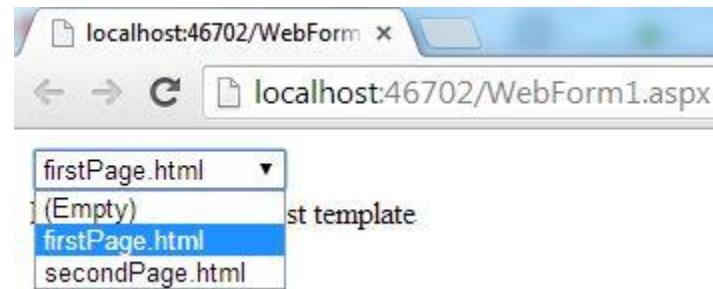
After this you can see two script tags in which two templates are provided, as in my previous article I told you that for creating the template you need to use the ng-template; you can see that I have also used that.

Now our application is created and is ready for execution.

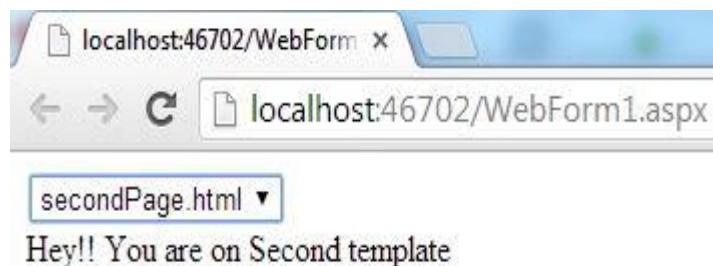
Output: On running the application you will see a drop down in which the first URL will be checked by default:



Now if I click on the drop down then all of the Template URL will be shown:



I have selected the second URL and you can see that different data is shown:



The complete code of this application is as follows:

```

<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.templatePages =
                [{ temp: 'firstPage.html', url: 'firstPage.html' }
                , { temp: 'secondPage.html', url: 'secondPage.html' }];
            $scope.tempPage = $scope.templatePages[0];
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">

```

```
<div ng-app>
  <div ng-controller="x">
    <select ng-model="tempPage" ng-options="p.temp for p in templatePages">
      <option value="">(Empty)</option>
    </select>
    <div ng-include="tempPage.url">
    </div>
  </div>
  <script type="text/ng-template" id="firstPage.html">
    Hey!! You are on First template
  </script>
  <script type="text/ng-template" id="secondPage.html">
    Hey!! You are on Second template
  </script>
  </div>
</form>
</body>
</html>
```

Create To Do List Using AngularJS

Introduction

This shows how to create a To Do List using AngularJS.

You can create many types of applications using AngularJS, here I show how to create another useful and easy to create application named "To Do List".

Using this application you can create a list of work that needs to be completed and can specify those works that are completed. This application will also show you how much work is still to be done.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the External JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the View/JavaScript of this application.

Add this code to the head section:

```
<script>
    function x($scope) {
        $scope.schedule = [
            { work: 'Wash the Car', completed: true },
```

```

{ work: 'Meet Ajay', completed: false }];

$scope.addNew = function () {
    $scope.schedule.push({ work: $scope.todoWork, completed: false });
    $scope.todoWork = "";
};

$scope.pending = function () {
    var count = 0;
    angular.forEach($scope.schedule, function (todo) {
        count += todo.completed ? 0 : 1;
    });
    return count;
};
}

</script>

```

Here I created three functions, the first is the main function to hold the two other functions and some initial values as well.

The second function is for adding new work to the To Do List.

The third function finds the pending work and counts how much is completed.

First I created a function named "x", in this function some initial values are declared under the variable "schedule".

After this a new function is created and named "addNew", this function will help to enter new work to the list.

At the end another function is created, "pending()"; this function will count the number of pending and total work and will show how much work is still pending of all the work.

Now our work on the View is completed and we can move to the ViewModel

Step 3 - Write this code in the body section:

```
<div ng-app>
  <h2>To Do List</h2>
  <div ng-controller="x">
    <span>{{pending()}} of {{schedule.length}} pending</span>
    <ul>
      <li ng-repeat="todo in schedule">
        <input type="checkbox" ng-model="todo.completed">
        <span class="completed-{{todo.completed}}">{{todo.work}}</span>
      </li>
    </ul>
    <input type="text" ng-model="todoWork" size="30" placeholder="add new work here">
    <input class="btn-primary" type="button" ng-click="addNew()" value="add">
  </div>
</div>
```

The function "x" is bound to a Div using the ng-controller.

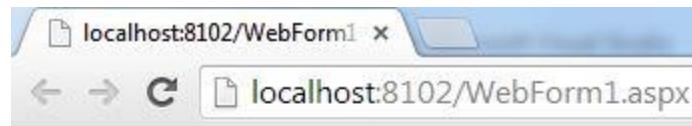
After this a span is used bound to the value of the pending and the total length of work.

A dynamic list is created using the ng-repeat directive.

At the end a TextBox and a button is taken, the user can enter new work in the TextBox and as he clicks on the button this new work will be entered into the list.

Now our application is created and is ready for execution.

Output: On running the application you will get an output like this one:



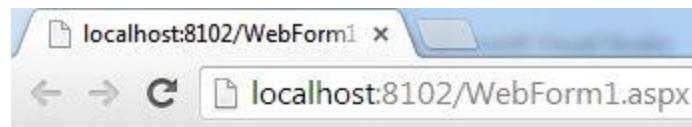
Todo List

1 of 2 pending

- Wash the Car
- Meet Ajay

Here by default two work is shown, one of these is checked and is showing as completed. You can also see the remaining work and total work is also displayed by this application.

Now I will enter a new work and then will click on the button to add it into the list.

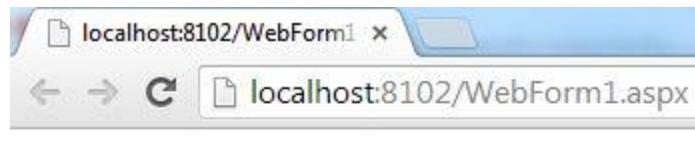


Todo List

1 of 2 pending

- Wash the Car
- Meet Ajay

Now as I click on the button the list will be updated along with the total number of work and the work that is still pending.



Todo List

2 of 3 pending

- Wash the Car
- Meet Ajay
- Call Shreya

The complete code of this application is as follows:

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <style>
        .completed-true {
            text-decoration: line-through;
            color: grey;
        }
    </style>
    <script>
        function x($scope) {
            $scope.schedule = [
                { work: 'Wash the Car', completed: true },
                { work: 'Meet Ajay', completed: false }];
            $scope.addNew = function () {
                $scope.schedule.push({ work: $scope.todoWork, completed: false });
                $scope.todoWork = "";
            };
            $scope.pending = function () {

```

```
var count = 0;
angular.forEach($scope.schedule, function (todo) {
    count += todo.completed ? 0 : 1;
});
return count;
};

}
</script>
</head>
<body>
<form id="form1" runat="server">
    <div ng-app>
        <h2>To Do List</h2>
        <div ng-controller="x">
            <span>{{pending()}} of {{schedule.length}} pending</span>
            <ul>
                <li ng-repeat="todo in schedule">
                    <input type="checkbox" ng-model="todo.completed">
                    <span class="completed-{{todo.completed}}">{{todo.work}}</span>
                </li>
            </ul>
            <input type="text" ng-model="todoWork" size="30" placeholder="add new work here">
            <input class="btn-primary" type="button" ng-click="addNew()" value="add">
        </div>
    </div>
</form>
</body>
</html>
```

Archive Completed Work From To Do List Using AngularJS

Introduction

In this I will tell you how to archive a completed work from the To Do List using AngularJS.

In previous I explained To Do List was created, but the work that was completed was not removed from the list, in this I will modify that application and will add an Archive Property by which only that work will be removed that is completed.

Using this application you can create a list of work that needs to be completed and can Archive those works that are completed. This application will also show you how much work is still to be done.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app to the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the View/JavaScript of this application.

Add this code to the head section:

```
<script>
function x($scope) {
    $scope.schedule = [
        { work: 'Wash the Car', completed: true },
        { work: 'Meet Ajay', completed: false }];
    $scope.addNew = function () {
        $scope.schedule.push({ work: $scope.todoWork, completed: false });
        $scope.todoWork = "";
    };
    $scope.pending = function () {
        var count = 0;
        angular.forEach($scope.schedule, function (todo) {
            count += todo.completed ? 0 : 1;
        });
        return count;
    };
    $scope.archive = function () {
        var oldTodos = $scope.schedule;
        $scope.schedule = [];
        angular.forEach(oldTodos, function (todo) {
            if (!todo.completed) $scope.schedule.push(todo);
        });
    };
}
</script>
```

I first created a function named "x", in this function some initial values are declared under a variable "schedule", among which one is tagged as completed.

After this a new function is created and named "addNew", this function will help to enter new work in the list.

The next function is named "pending()", this function will count the number of pending and the total work and will show how much work is still pending out of all the total work.

In the end our main function is created, in other words to archive the completed work, it is named archived.

In this function default values are first passed into a variable "oldTodos", after this it is checking whether or not any work is completed, if it is then it will be removed them from the list.

Now our work on the View is completed and we can proceed to the ViewModel.

Step 3 - Write this code in the body section:

```
<div ng-app>
  <h2>Todo List</h2>
  <div ng-controller="x">
    <span>{{pending()}} of {{schedule.length}} pending</span>
    [ <a href="" ng-click="archive()">archive</a> ]
    <ul>
      <li ng-repeat="todo in schedule">
        <input type="checkbox" ng-model="todo.completed">
        <span class="completed-{{todo.completed}}">{{todo.work}}</span>
      </li>
    </ul>
    <input type="text" ng-model="todoWork" size="30" placeholder="add new work here">
    <input class="btn-primary" type="button" ng-click="addNew()" value="add">
  </div>
</div>
```

The function "x" is bound to a Div using ng-controller.

After this a span is taken that is bound to the value of pending and the total length of work.

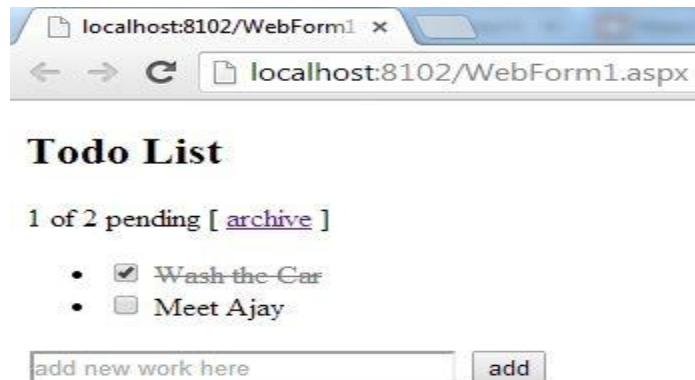
After this an anchor is taken whose click is bound with the archive() function, so the click of this anchor will remove the completed work.

A dynamic list is created using the ng-repeat directive..

At the end a TextBox and a button is taken, the user can enter new work in the TextBox and as he clicks on the button this new work will be entered into the list.

Now our application is created and is ready for execution.

Output: On running the application you will get an output like this one:



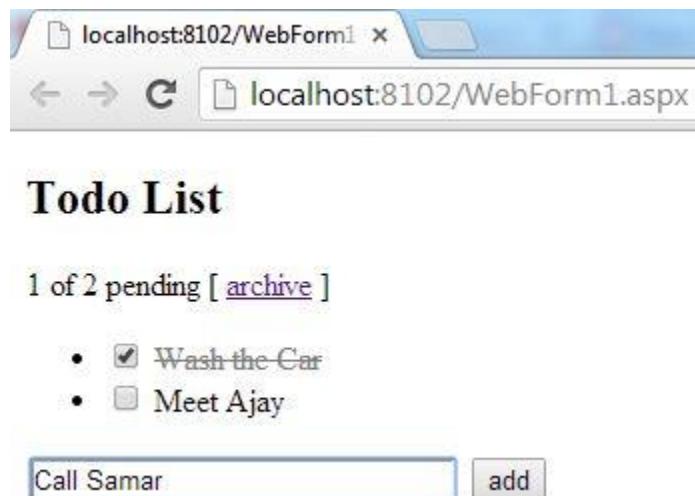
Todo List

1 of 2 pending [[archive](#)]

- Wash the Car
- Meet Ajay

Here by default two work is shown, one of these is checked and is showing as completed. You can also see the remaining work and total work are also displayed by this application.

Now I will enter a new work and then will click on the button to add it into the list.

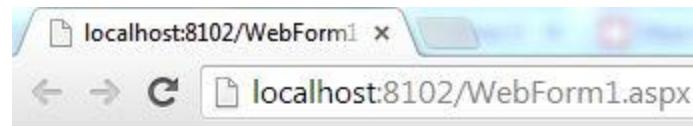


Todo List

1 of 2 pending [[archive](#)]

- Wash the Car
- Meet Ajay

Now as I click on the button the list will be updated along with the total amount of work and the work that is still pending.

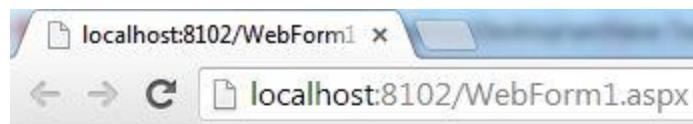


Todo List

1 of 3 pending [[archive](#)]

- Wash the Car
- Meet Ajay
- Call Samar

Now I am making this new work as completed work and then I will click on "Archive".



Todo List

1 of 1 pending [[archive](#)]

- Meet Ajay

You can see that the click of the Anchor has removed the Completed work.

The complete code of this application is as follows:

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<script src="angular.min.js"></script>
<style>
.completed-true {
    text-decoration: line-through;
    color: grey;
}
</style>
<script>
function x($scope) {
    $scope.schedule = [
        { work: 'Wash the Car', completed: true },
        { work: 'Meet Ajay', completed: false }];

    $scope.addNew = function () {
        $scope.schedule.push({ work: $scope.todoWork, completed: false });
        $scope.todoWork = "";
    };

    $scope.pending = function () {
        var count = 0;
        angular.forEach($scope.schedule, function (todo) {
            count += todo.completed ? 0 : 1;
        });
        return count;
    };
    $scope.archive = function () {
        var oldTodos = $scope.schedule;
        $scope.schedule = [];
        angular.forEach(oldTodos, function (todo) {
            if (!todo.completed) $scope.schedule.push(todo);
        });
    };
}

```

```
</script>
</head>
<body>
<form id="form1" runat="server">
<div ng-app>
    <h2>Todo List</h2>
    <div ng-controller="x">
        <span>{{pending()}} of {{schedule.length}} pending</span>
        [ <a href="" ng-click="archive()">archive</a> ]
        <ul>
            <li ng-repeat="todo in schedule">
                <input type="checkbox" ng-model="todo.completed">
                <span class="completed-{{todo.completed}}">{{todo.work}}</span>
            </li>
        </ul>
        <input type="text" ng-model="todoWork" size="30" placeholder="add new work here">
        <input class="btn-primary" type="button" ng-click="addNew()" value="add">
    </div>
</div>
</form>
</body>
</html>
```

Show Data in Dynamic Grid Using AngularJS

Introduction

This explains how to show data in a dynamic grid using AngularJS.

AngularJS provides many features for creating various kinds of applications. Here I will create an application with a dynamic grid using AngularJS.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will first work on the ViewModel or JavaScript part of this application. So, write this code in the head section of your application:

```
<script>
    var empid = 1;
    function x($scope) {
        $scope.employees = [
            { id: 0, 'name': 'Anubhav', 'address': 'Ghaziabad', 'dept': 'Developer' }
        ];

        $scope.saveRecord = function () {
            if ($scope.newEmployee.id == null) {
```

```

$scope.newEmployee.id = empid++;
$scope.employees.push($scope.newEmployee);
} else {

    for (i in $scope.employees) {
        if ($scope.employees[i].id == $scope.newEmployee.id) {
            $scope.employees[i] = $scope.newEmployee;
        }
    }
}
$scope.newEmployee = {};
}
</script>

```

Here first I have created an id that will increase the number of employees to be entered, it's initial value is set to 1.

Then a function is created, the function "x"; this function will be bound to the controller. In this function a variable, "employees", is used; in this variable some initial values about the first employee is passed like it's name, address and so on.

After this another function is created for adding the new employees, it will check whether or not the id is null, if it's not null then the employee Id will be increased by one and a new entry will be made in the employee list.

Now our work on the ViewModel is completed and we can move to the View part, the design part.

Step 3 - Write this code in the body section of your application:

```

<body>
<form id="form1" runat="server">
    <div ng-app="" ng-controller="x">
        <label>Name</label>
        <input type="text" name="name" ng-model="newEmployee.name"/>
        <label>Address</label>
        <input type="text" name="address" ng-model="newEmployee.address"/>
        <label>Dept.</label>
    </div>
</form>

```

```

<input type="text" name="dept" ng-model="newEmployee.dept"/>
<br/>
<input type="hidden" ng-model="newEmployee.id" />
<input type="button" value="Save" ng-click="saveRecord()"/>
<table>
  <tr>
    <th>Name</th>
    <th>Address</th>
    <th>Dept</th>
  </tr>
  <tr ng-repeat="employee in employees">
    <td>{{ employee.name }}</td>
    <td>{{ employee.address }}</td>
    <td>{{ employee.dept }}</td>
  </tr>
</table>
</div>
</form>
</body>

```

Here the parent div is bound to the function "x" using the ng-controller directive.

In this div three TextBoxes are used, the first TextBox is bound to the name of the employee, the second TextBox is bound to the address and the last one is bound to the department of the Employee. All these TextBoxes are bound using the ng-model directive.

After the TextBoxes a button is added with a click firing the function "saveRecord".

After this a table is added with a first row showing the column headings and the second row is bound using the ng-repeat, so this is a dynamic row that will increase by every new entry made to the table.

In this row three columns are created that are bound to the employee's name, address and department. Since these are part of the dynamic columns they will also be automatic on each new entry made.

Now our application is created and is ready for execution.

Output: On running the application you will get output like this:

localhost:8102/WebForm1 x localhost:8102/WebForm1.aspx

Name	Address	Dept.
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Save"/>		

Name	Address	Dept
Amubhav	Ghaziabad	Developer

Here you can see that by default one row of data is shown in the grid; that's because we have provided this data as the initial value.

Now I am entering a new row of data.

localhost:8102/WebForm1 x localhost:8102/WebForm1.aspx

Name	Address	Dept.
<input type="text" value="Mohit"/>	<input type="text" value="Noida"/>	<input type="text" value="Designer"/>
<input type="button" value="Save"/>		

Name	Address	Dept
Amubhav	Ghaziabad	Developer

Now as I click on the button then this new row of data will be entered into the previously existing Grid.

localhost:8102/WebForm1 x localhost:8102/WebForm1.aspx

Name	Address	Dept.
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Save"/>		

Name	Address	Dept
Amubhav	Ghaziabad	Developer
Mohit	Noida	Designer

You can see that my Grid is updated.

The complete code of this application is as follows:

```

<body>
    <form id="form1" runat="server">
        <div ng-app="" ng-controller="x">
            <label>Name</label>
            <input type="text" name="name" ng-model="newEmployee.name"/>
            <label>Address</label>
            <input type="text" name="address" ng-model="newEmployee.address"/>
            <label>Dept.</label>
            <input type="text" name="dept" ng-model="newEmployee.dept"/>
            <br/>
            <input type="hidden" ng-model="newEmployee.id" />
            <input type="button" value="Save" ng-click="saveRecord()" class="btn btn-primary"/>
            <br />
            <br />
            <table border="1" bordercolor="blue">
                <tr style="color:blue">
                    <th style="width:150px">Name</th>
                    <th style="width:150px">Address</th>
                    <th style="width:150px">Dept</th>
                </tr>
                <tr style="color:pink" ng-repeat="employee in employees">
                    <td>{{ employee.name }}</td>
                    <td>{{ employee.address }}</td>
                    <td>{{ employee.dept }}</td>
                </tr>
            </table>
        </div>
    </form>
</body>

```

Apply CRUD Operations in Dynamic Grid Using AngularJS

Introduction

In this I will tell you how to apply CRUD operations in a Dynamic Grid using AngularJS.

Previously I showed you how to show the data in Grid Format but in this article we will move one step further and make the Grid editable, in other words today I will provide CRUD operations for the Grid.

Step 1 - Since I am working on a previous application you can check the previous article to see at which step I had left previously.

Now I will update the JavaScript section and will provide the edit and delete functionality. For that you need to write this code in the script section:

```
$scope.delete = function (id) {
    for (i in $scope.employees) {
        if ($scope.employees[i].id == id) {
            $scope.employees.splice(i, 1);
            $scope.newEmployee = {};
        }
    }
}
$scope.edit = function (id) {
    for (i in $scope.employees) {
        if ($scope.employees[i].id == id) {
            $scope.newEmployee = angular.copy($scope.employees[i]);
        }
    }
}
```

First I provided the delete functionality, for this I have created a function named "delete". In this function the Id of the Employee is passed, if the Id correctly matches one of the existing Ids then the delete operation will be done and that entry will be deleted.

Similarly, I have created the "Edit" function. In this function the Id of the Employee will also be passed but this time the data related to that Id will not be deleted, instead it will be passed back to the corresponding textboxes so that the user can make changes and when the user clicks the save button the data will again be saved at the same Id position as it was stored previously.

Now the updated Script looks like this:

```
<script>
    var empid = 1;
    function x($scope) {

        $scope.employees = [
            { id: 0, 'name': 'Anubhav', 'address': 'Ghaziabad', 'dept': 'Developer' }
        ];
        $scope.saveRecord = function () {
            if ($scope.newEmployee.id == null) {
                $scope.newEmployee.id = empid++;
                $scope.employees.push($scope.newEmployee);
            } else {

                for (i in $scope.employees) {
                    if ($scope.employees[i].id == $scope.newEmployee.id) {
                        $scope.employees[i] = $scope.newEmployee;
                    }
                }
                $scope.newEmployee = {};
            }
        }
        $scope.delete = function (id) {

            for (i in $scope.employees) {
                if ($scope.employees[i].id == id) {
                    $scope.employees.splice(i, 1);
                    $scope.newEmployee = {};
                }
            }
        }
    }
}
```

```
$scope.edit = function (id) {
    for (i in $scope.employees) {
        if ($scope.employees[i].id == id) {
            $scope.newEmployee = angular.copy($scope.employees[i]); } } }
</script>
```

Step 2 - Now I will work on the ViewModel of this application and will modify it as well.

You need to update your Table with this code:

```
<table border="1" bordercolor="blue">
<tr style="color:blue">
<th style="width:150px">Name</th>
<th style="width:150px">Address</th>
<th style="width:150px">Dept</th>
<th>Action</th>
</tr>
<tr style="color:pink" ng-repeat="employee in employees">
<td>{{ employee.name }}</td>
<td>{{ employee.address }}</td>
<td>{{ employee.dept }}</td>
<td>
<a href="#" ng-click="edit(employee.id)">edit</a> |
<a href="#" ng-click="delete(employee.id)">delete</a>
</td>
</tr>
</table>
```

Here in the first Row I added one more heading, "Action". In the second row I added one more column in which two Anchors are used.

The first Anchor click is bound to the "Edit" function and the second Anchor click is bound to the "Delete" function.

The Id of the Employee is passed depending on the Anchors, in other words wherever the Anchors are placed, the corresponding Id will be passed to the function.

Now our View Model is updated and it's code is like this:

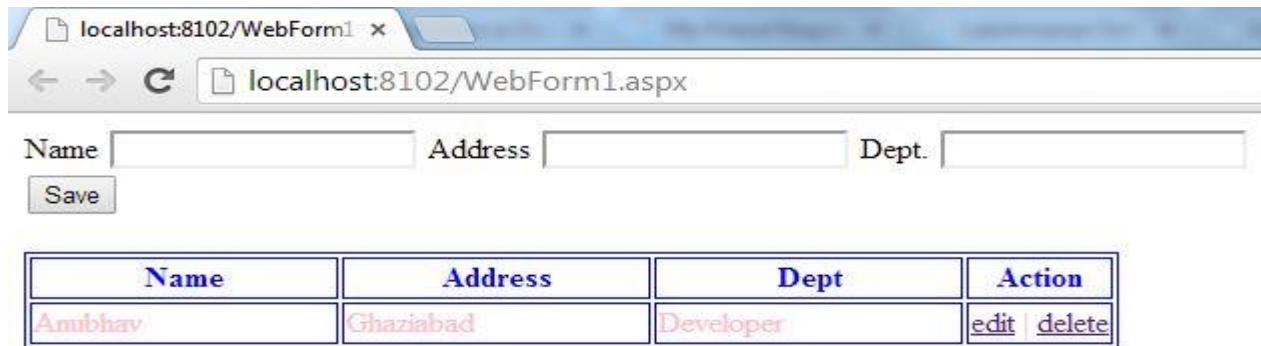
```

<div ng-app="" ng-controller="x">
    <label>Name</label>
    <input type="text" name="name" ng-model="newEmployee.name"/>
    <label>Address</label>
    <input type="text" name="address" ng-model="newEmployee.address"/>
    <label>Dept.</label>
    <input type="text" name="dept" ng-model="newEmployee.dept"/>
    <br/>
    <input type="hidden" ng-model="newEmployee.id" />
    <input type="button" value="Save" ng-click="saveRecord()" class="btn btn-primary"/>
    <br />
    <br />
    <table border="1" bordercolor="blue">
        <tr style="color:blue">
            <th style="width:150px">Name</th>
            <th style="width:150px">Address</th>
            <th style="width:150px">Dept</th>
            <th>Action</th>
        </tr>
        <tr style="color:pink" ng-repeat="employee in employees">
            <td>{{ employee.name }}</td>
            <td>{{ employee.address }}</td>
            <td>{{ employee.dept }}</td>
            <td>
                <a href="#" ng-click="edit(employee.id)">edit</a> |
                <a href="#" ng-click="delete(employee.id)">delete</a>
            </td>
        </tr>
    </table>
</div>

```

Now our application is created and is ready for execution.

Output: On running the application you will get output like this:

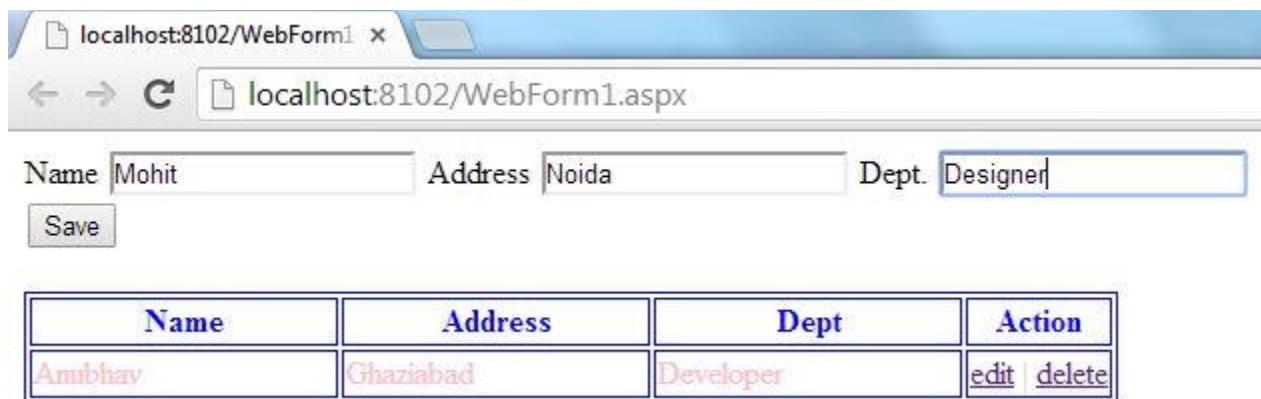


The screenshot shows a web page at localhost:8102/WebForm1.aspx. At the top, there are input fields for Name, Address, and Dept., followed by a Save button. Below these is a table with columns: Name, Address, Dept, and Action. A single row contains the values Amibhav, Ghaziabad, Developer, and edit | delete.

Name	Address	Dept	Action
Amibhav	Ghaziabad	Developer	edit delete

Here you can see that a default row of data is available in the grid along with Edit and Delete buttons.

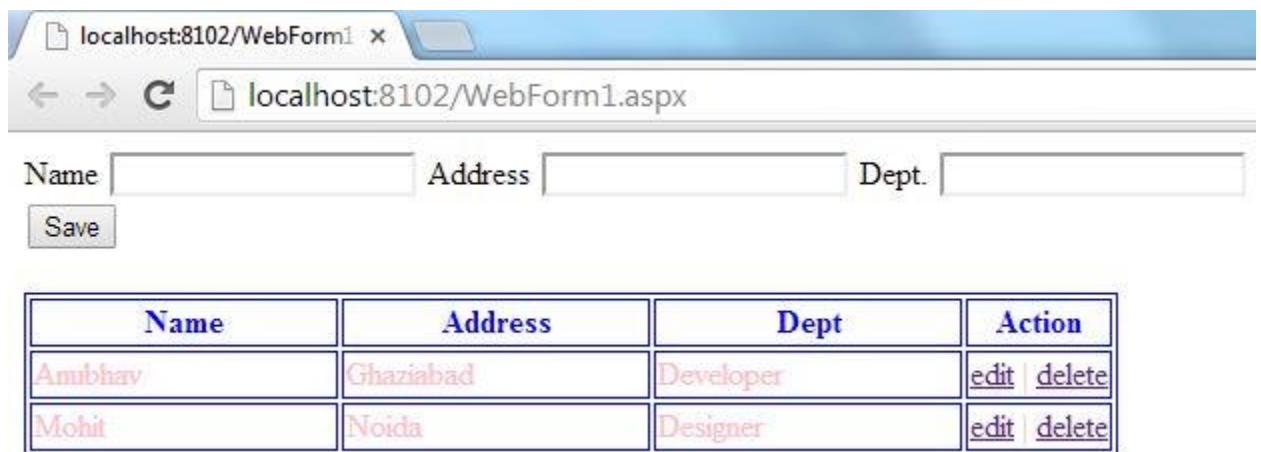
Now, I am adding a new row of data.



The screenshot shows the same web page after entering new data. The Name field now contains "Mohit", the Address field contains "Noida", and the Dept. field contains "Designer". The Save button is visible below the inputs. The grid has been updated to reflect the new row.

Name	Address	Dept	Action
Amibhav	Ghaziabad	Developer	edit delete
Mohit	Noida	Designer	

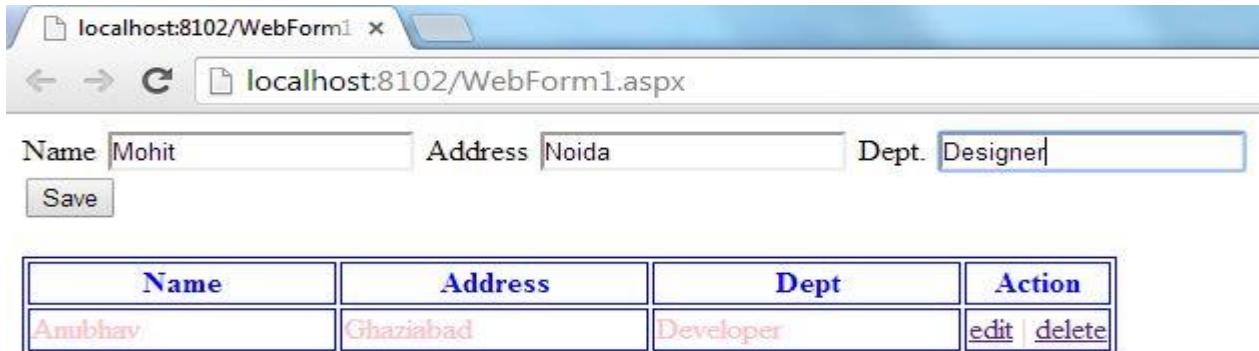
As I click on the Save button this new data will be entered into the Grid.



The screenshot shows the grid after the "Save" button was clicked. The new row with "Mohit" and "Noida" has been successfully added to the database, appearing in the grid along with the previous row.

Name	Address	Dept	Action
Amibhav	Ghaziabad	Developer	edit delete
Mohit	Noida	Designer	edit delete

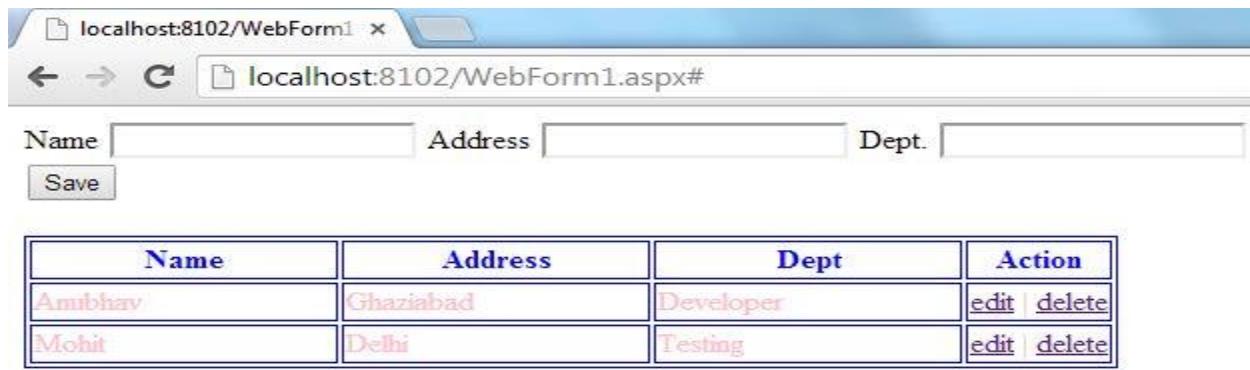
Now I will click on Edit button and the data can again be seen in the textboxes



The screenshot shows a web page with a header bar containing the URL "localhost:8102/WebForm1.aspx". Below the header is a form with three text input fields: "Name" (containing "Mohit"), "Address" (containing "Noida"), and "Dept." (containing "Designer"). Below the form is a "Save" button. Underneath the form is a table with four columns: "Name", "Address", "Dept.", and "Action". The first row of the table contains "Anubhav", "Ghaziabad", "Developer", and edit|delete buttons. The second row contains "Mohit", "Delhi", "Testing", and edit|delete buttons.

Name	Address	Dept.	Action
Anubhav	Ghaziabad	Developer	edit delete
Mohit	Delhi	Testing	edit delete

Now I will update this data and will again click on the save button.



The screenshot shows a web page with a header bar containing the URL "localhost:8102/WebForm1.aspx#". Below the header is a form with three text input fields: "Name" (empty), "Address" (empty), and "Dept." (empty). Below the form is a "Save" button. Underneath the form is a table with four columns: "Name", "Address", "Dept.", and "Action". The first row of the table contains "Anubhav", "Ghaziabad", "Developer", and edit|delete buttons. The second row contains "Mohit", "Delhi", "Testing", and edit|delete buttons.

Name	Address	Dept.	Action
Anubhav	Ghaziabad	Developer	edit delete
Mohit	Delhi	Testing	edit delete

You can see that the data is modified.

Now I will click on the Delete Button at the first row of data and will check what has happened.



The screenshot shows a web page with a header bar containing the URL "localhost:8102/WebForm1.aspx#". Below the header is a form with three text input fields: "Name" (empty), "Address" (empty), and "Dept." (empty). Below the form is a "Save" button. Underneath the form is a table with four columns: "Name", "Address", "Dept.", and "Action". The second row contains "Mohit", "Delhi", "Testing", and edit|delete buttons.

Name	Address	Dept.	Action
Mohit	Delhi	Testing	edit delete

You can see that only the second row of data is available.

Using angular.forEach API in AngularJS

Introduction

This explains the "angular.forEach" API of AngularJS.

AngularJS provides many types of directives and an API that helps us greatly to solve our issues in a very significant manner, here I will also show one more API provided by AngularJS, "angular.forEach".

I will create a simple application that will help you in an easy way to understand How you can use it in your Complex Applications.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript section of this application, you need to add the following code to the head section of your application:

```
<script>
    function x($scope) {
        $scope.names = [{ Name: 'Anubhav' , Male:true},
                      { Name: 'Mohit' , Male: true },
                      { Name: 'Komal' , Male: false },
```

```

{ Name: 'Anupriya', Male: false },
{ Name: 'Girish', Male: true }];
$scope.func = function () {
    var male = 0;
    angular.forEach($scope.names, function (todo) {
        male += todo.Male ? 1 : 0;
    });
    return male;
};
</script>

```

Here I have created a function "x" that will be bound as the Controller in the design part of our application.

In this function I have provided some initial values under the Name and their Gender.

After this I had again created a function "func", in this function I initialized a variable, "male", whose value is set to 0.

After initializing the function I used the "angular.forEach()" API. Using angular.forEach we will check whether a person is Male or Female, if male then the variable "male" will increase it's value by one and in this way all the users and their genders will be checked and finally the output will be provided.

Now our work with JavaScript is completed and we can move to the design part.

Step 3 - Write this code in the body section of your application:

```

<body>
<form id="form1" runat="server">
<div ng-app ng-controller="x">
<h2>New Development Team</h2>
<ul>
<li ng-repeat="todo in names">
<input ng-model="todo.Name" />

```

```

<input type="checkbox" ng-model="todo.Male" value="Male" />Male
</li>
</ul>
<span>{{func()}} Male Members are in our Team</span>
</div>
</form>
</body>

```

Here I bound the Div with the function "x" using the ng-controller directive.

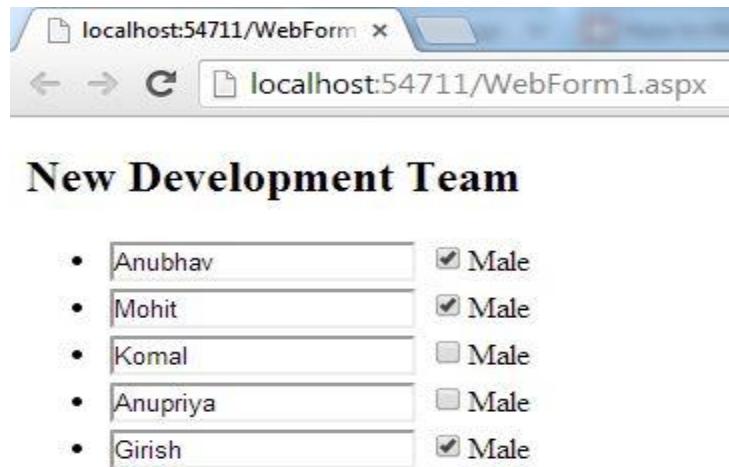
In this Div I created an unordered list that is made dynamic using the ng-repeat directive. The ng-repeat Directive helps to create things dynamic, in other words it will work for all the initial values that are provided in the Script section and will allow us to not use multiple Textboxes or Spans to show the data.

In this List I took a TextBox and a checkbox, the TextBox is bound to the Names available in the script using ng-model and the Checkbox is bound to the Male/Female value again by using the ng-model.

At the end I used a span bound to the function "func", so this span will show the output provided by the "func".

Now our application is created and is ready for execution.

Output: On running the application you will get output like this:



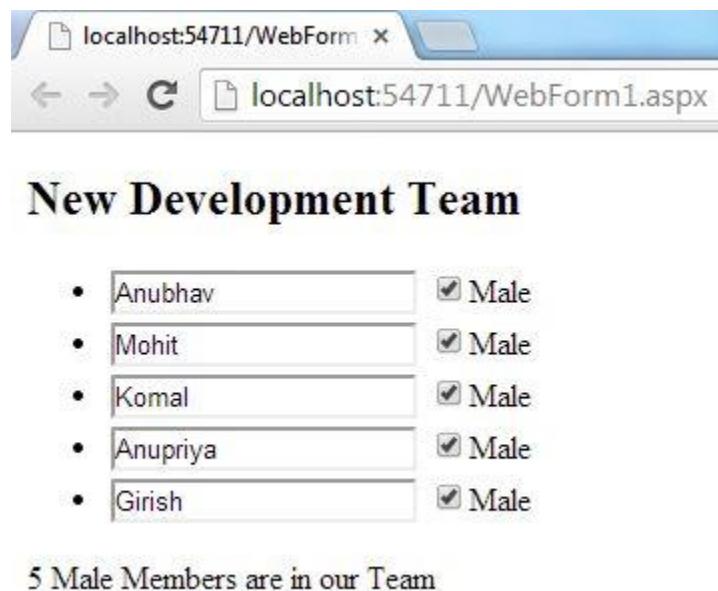
• Anubhav	<input checked="" type="checkbox"/> Male
• Mohit	<input checked="" type="checkbox"/> Male
• Komal	<input type="checkbox"/> Male
• Anupriya	<input type="checkbox"/> Male
• Girish	<input checked="" type="checkbox"/> Male

3 Male Members are in our Team

You can see that all the Initial values are displayed in the Textboxes and Male/Female is displayed in the corresponding Checkbox.

Number of Mail members is also displayed at the end.

Now if I make changes in this output and make some people Female to Male or Vice Versa then the total number of males will also be changed



New Development Team

- Male
- Male
- Male
- Male
- Male

5 Male Members are in our Team

The complete code of this application is as follows:

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<script src="angular.min.js"></script>
<script>
function x($scope)
{
    $scope.names =
[ { Name: 'Anubhav' , Male:true},
```

```
{ Name: 'Mohit', Male: true },
{ Name: 'Komal', Male: false },
{ Name: 'Anupriya', Male: false },
{ Name: 'Girish', Male: true }];
$scope.func = function () {
    var male = 0;
    angular.forEach($scope.names, function (todo)
    {
        male += todo.Male ? 1 : 0;
    });
    return male;
};
</script>
</head>
<body>
<form id="form1" runat="server">
<div ng-app ng-controller="x">
    <h2>New Development Team</h2>
    <ul>
        <li ng-repeat="todo in names">
            <input ng-model="todo.Name" />
            <input type="checkbox" ng-model="todo.Male" value="Male" />Male
        </li>
    </ul>
    <span>{{func()}} Male Members are in our Team</span>
</div>
</form>
</body>
</html>
```

Create File Uploader Using AngularJS

Introduction

In this I will tell you how to create a file uploader using AngularJS.

You might have used a file uploader of ASP.NET, but what about a file uploader in AngularJS? You might have been using AngularJS in your application and if such type of application can be provided then it is just a bonus to us.

For providing this type of functionality Angular uses "angularFileUpload" and \$upload.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript section of this application.

Write this code in the head section:

```

<script>
    angular.module('mod', ['angularFileUpload']);
    var x = ['$scope', '$upload', function ($scope, $upload)
    {
        $scope.selectFile = function ($slctfl)
        {
            $upload.upload
            ({
                url: 'my/upload/url'  }) } }];
</script>

```

Here I have created angular.Module in which "mod" and "angularFileUpload" are defined, mod is the variable that will help to bind using the ng-app.

After this I have created a function "x", this function will be bound to the body section using the ng-controller directive. ng-app and ng-directive is among the most important directives when using AngularJS.

In this function I used a variable named "selectFile", in this function the upload directive is used with the property named URL. This URL will follow the path that will be provided by the user.

Just this much code is necessary to do our work of uploading a file for the user. Now we can move to the design part of this application.

Step 3 - Write this code in the body section:

```

<body>
    <form id="form1" runat="server">
        <div ng-app="mod" ng-controller="x">
            <input type="file" ng-file-select="selectFile($slctfl)" />
        </div>
    </form>
</body>

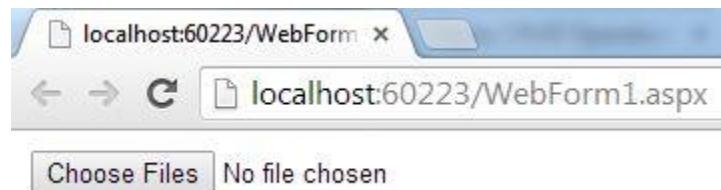
```

Here I had bound the Div using the ng-app directive.

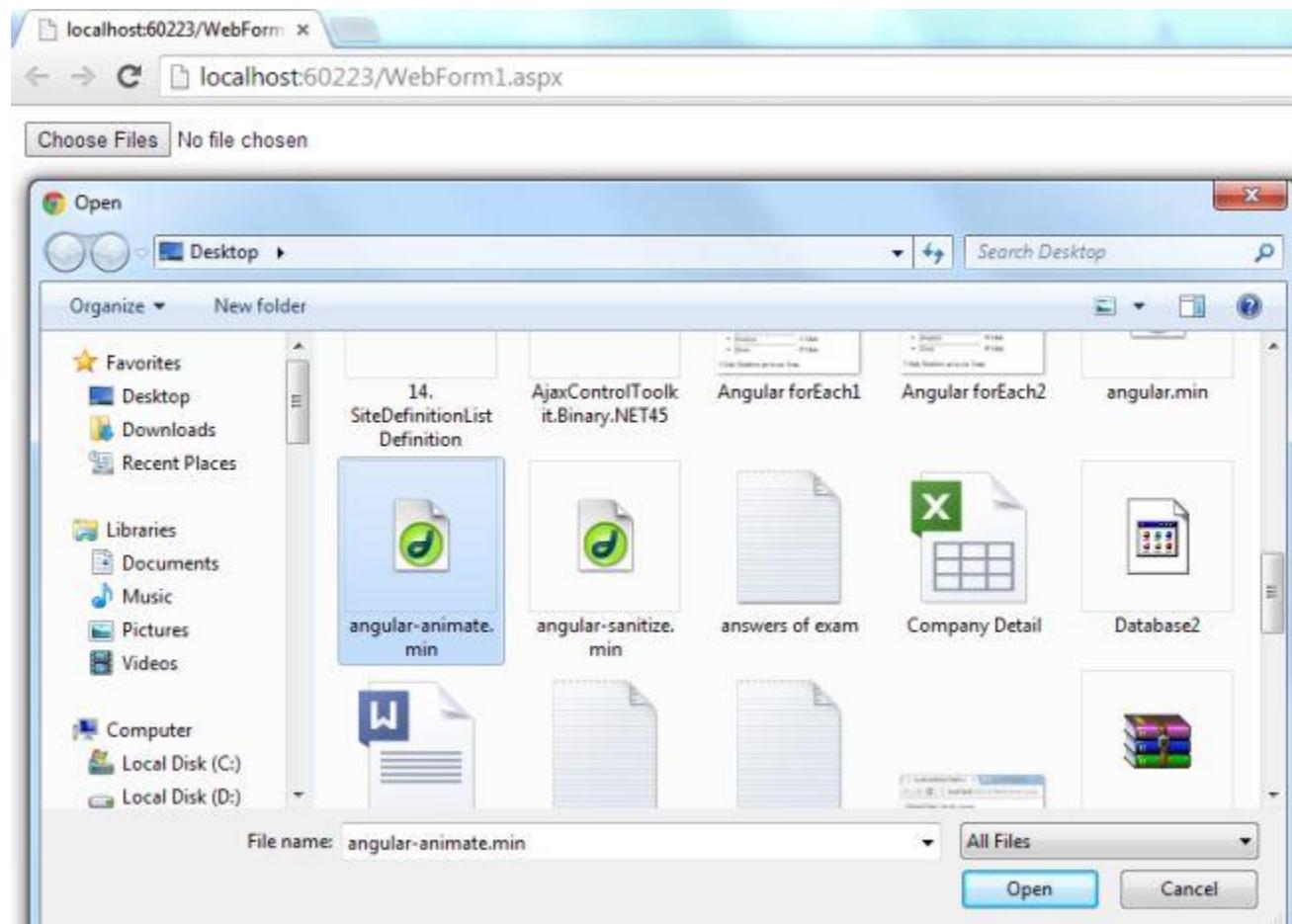
In this Div I took an input tag whose type is set to "File", this input tag is bound using the ng-file-select directive to the function "selectFile(\$slctfl)".

Now our application is created and is ready for execution.

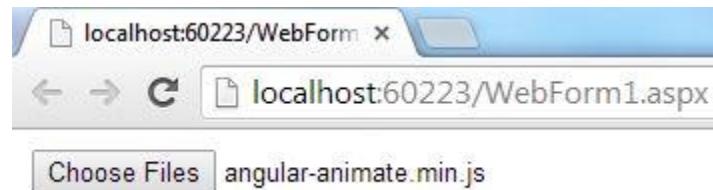
Output: On running the application you will see only a button in the output window:



If I click on the button then a new window will be opened that will allow me to select a file:



On selecting the file I clicked on the open button and you can see that the file is uploaded:



The complete code of this application is as follows:

```
<html ng-app="mod" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        angular.module('mod', ['angularFileUpload']);

        var x = ['$scope', '$upload', function ($scope, $upload) {
            $scope.selectFile = function ($slctfl) {
                $upload.upload({
                    url: 'my/upload/url'
                })
            }
        }];
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-app="mod" ng-controller="x">
            <input type="file" ng-file-select="selectFile($slctfl)" multiple>
        </div>
    </form>
</body>
</html>
```

Create Multi File Uploader Using AngularJS

Introduction

In this I will tell you how to create a multiple file uploader using AngularJS.

Previously I showed you, an application where I created a file uploider using AngularJS, but using that uploader you can upload only one file, today I am modifying that uploader and making a multiple file uploader.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript for this application.

Now you need to modify your script with this code:

```
<script>
    angular.module('mod', ['angularFileUpload']);

    var x = ['$scope', '$upload', function ($scope, $upload) {
        $scope.selectFile = function ($slctfl) {
```

```

for (var i = 0; i < $slctfl.length; i++) {
    var $fl = $slctfl[i];
    $upload.upload({
        url: 'my/upload/url',
        file: $fl,
        progress: function (e) {}
    }).then(function (data, status, headers, config) {
        console.log(data); });
}
</script>

```

Here I created angular.Module in which "mod" and "angularFileUpload" are defined, mod is the variable that will help to bind using the ng-app.

After this I created a function "x", this function will be bound to the body section using the ng-controller directive. ng-app and ng-directive are among the most important directives when using AngularJS.

In this function I used a variable named "selectFile", in this function the for loop is executed that will execute until all the selected files are not uploaded, for this to happen the upload directive is used that has three properties named url, file and progress. This URL will follow the path that will be provided by the user.

Step 3 - Our work on the JavaScript is completed and we can proceed to the design part of this application.

Write this code in the body section:

```

<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <input type="file" ng-file-select="selectFile($slctfl)" multiple>
        </div>
    </form>
</body>

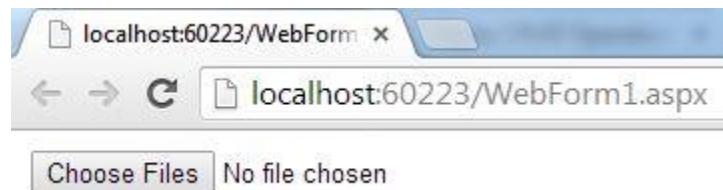
```

Here I bound the Div using the ng-app directive.

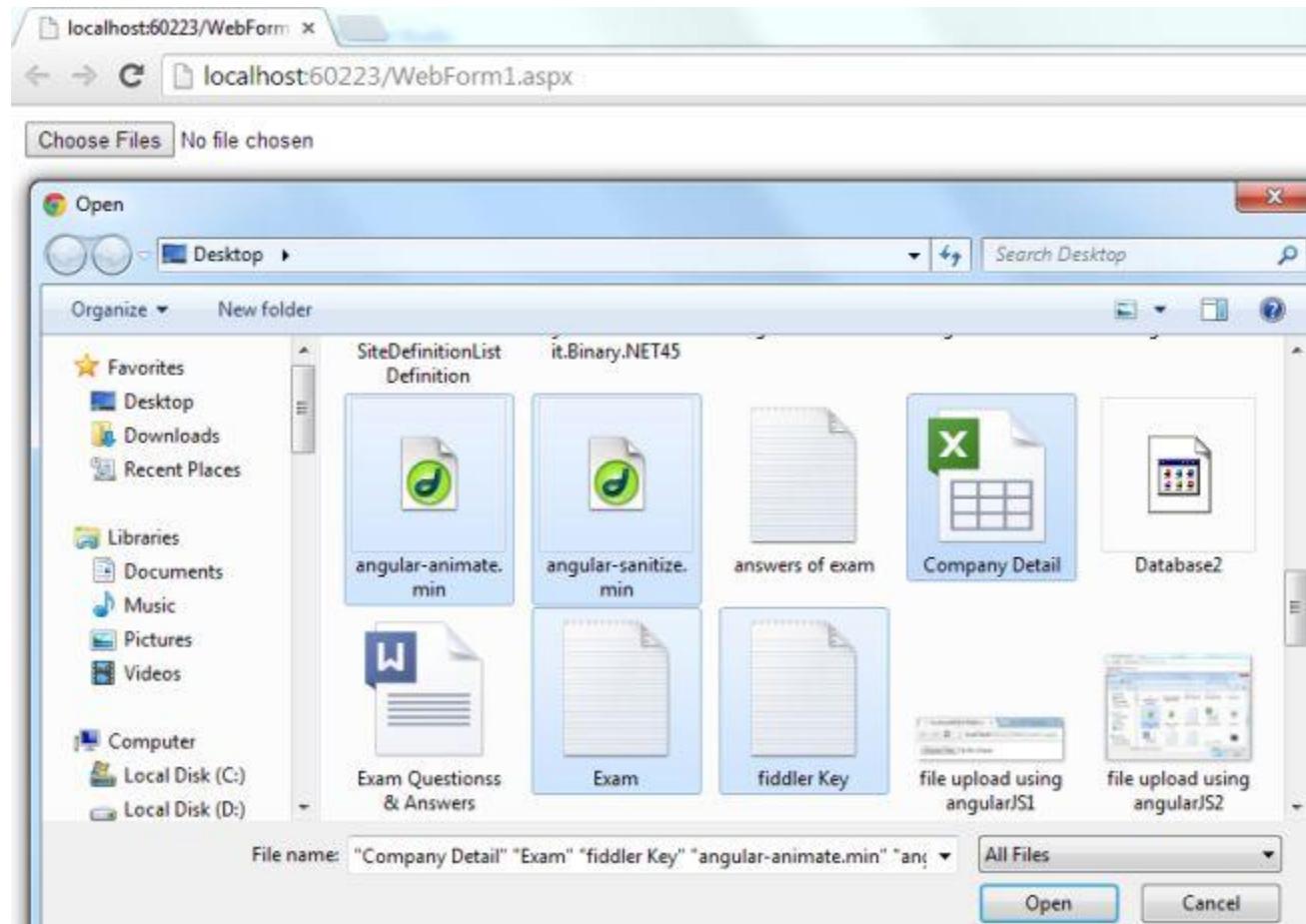
In this Div I took an input tag whose type is set to File, this input tag is bound using the ng-file-select Directive to the function "selectFile(\$slctfl)".

Now our application is created and is ready for execution.

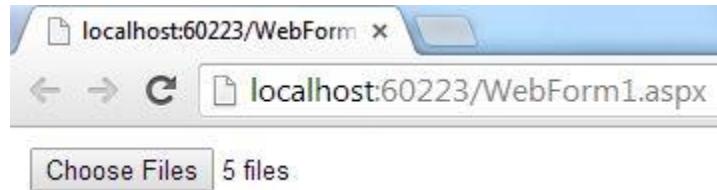
Output: On running the application you will see only a button in the output window:



If I click on the button then a new window will be opened that will allow me to select multiple files:



On selecting the files I click on the open button and you can see that a number of files are uploaded and shown as the total number of files:



The complete code of this application is as follows:

```

<html ng-app="mod" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        angular.module('mod', ['angularFileUpload']);

        var x = ['$scope', '$upload', function ($scope, $upload) {
            $scope.selectFile = function ($slctfl) {
                for (var i = 0; i < $slctfl.length; i++) {
                    var $fl = $slctfl[i];
                    $upload.upload({
                        url: 'my/upload/url',
                        file: $fl,
                        progress: function (e) { }
                    }).then(function (data, status, headers, config) {
                        console.log(data);
                    });
                }
            };
        }];
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <input type="file" ng-file-select="selectFile($slctfl)" multiple>
        </div>
    </form>
</body>
</html>

```

Use \$filter Service in AngularJs

Introduction

In this I will tell you how to use the \$filter service in AngularJs.

The \$filter service can be used to filter the data, it is declared along with the \$scope service.

Here I will create a simple example on \$filter that will help you understand how to use the \$filter service.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
  <title></title>
  <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file, the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application that will help you understand this directive.

First I will create a JavaScript function where \$filter will be declared and some initial values will be passed.

```
<script>
  function x($scope, $filter) {
    $scope.cars = [
      { carId: 001, carName: 'Santro' },
```

```

{ carId: 002, carName: 'i10 Grand' },
{ carId: 003, carName: '120' },
{ carId: 004, carName: 'Verna' },
{ carId: 005, carName: 'City' }];

$scope.cars2 = $scope.cars;

$scope.$watch('search', function (val) {
    $scope.cars = $filter('filter')($scope.cars2, val);
});

</script>

```

As I described earlier, \$filter is a service declared as \$scope is declared, you can see that I created a function "x" in which the \$scope and \$filter services are declared.

Now in this function a variable is declared named "cars", using this variable some initial values are passed in carId and carName.

At the end a filter is applied on these initial values.

Step 3 - Now our work on JavaScript is completed and we can move towards the design part of this application.

Write this code in the body section:

```

<div ng-app>
<div ng-controller="x">
    <input type="text" ng-model="search">
    <table>
        <tbody>
            <tr ng-repeat="car in cars">
                <td>{{car.carId}}</td>
                <td>{{car.carName}}</td>
            </tr>
        </tbody>
    </table>
</div>
</div>

```

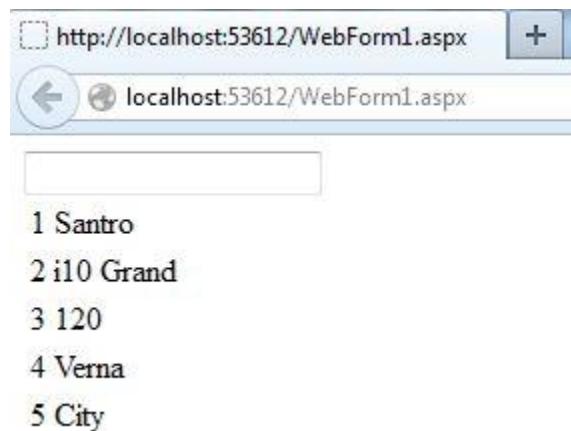
Here I had bound a div with the function "x" using the ng-controller.

After this a TextBox is created that is bound with the search using the ng-model. After this a table is created whose row is bound using the ng-repeat directive. This means that it will work as a dynamic table that will increase its row and column automatically depending on the data available.

The column of this table is bound with the Id and Name of the cars.

Now our application is created and is ready for execution.

Output: On running the application you will get output like this

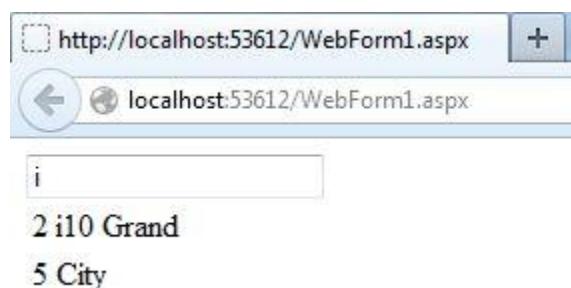


A screenshot of a web browser window. The address bar shows the URL `http://localhost:53612/WebForm1.aspx`. Below the address bar is a search bar containing the letter 'i'. Underneath the search bar is a table with five rows, each containing a number and a car name. The rows are: 1 Santro, 2 i10 Grand, 3 120, 4 Verna, and 5 City.

1	Santro
2	i10 Grand
3	120
4	Verna
5	City

Here you can see that a list of cars is displayed in the output, also a TextBox is available that can be used to filter the data.

Now I will enter a letter and whosoever has that letter in the car name will get filtered and will be displayed.



A screenshot of a web browser window, identical to the one above but with a different search term. The search bar now contains the letter 'i'. The table below shows only two rows that contain the letter 'i': 2 i10 Grand and 5 City.

2	i10 Grand
5	City

The complete code of this application is as follows:

```

<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope, $filter) {
            $scope.cars = [
                { carId: 001, carName: 'Santro' },
                { carId: 002, carName: 'i10 Grand' },
                { carId: 003, carName: '120' },
                { carId: 004, carName: 'Verna' },
                { carId: 005, carName: 'City' }];
            $scope.cars2 = $scope.cars;

            $scope.$watch('search', function (val) {
                $scope.cars = $filter('filter')($scope.cars2, val);
            });
        }
    </script> </head>
<body>
    <form carId="form1" runat="server">
        <div ng-app>
            <div ng-controller="x">
                <input type="text" ng-model="search">
                <table>
                    <tbody>
                        <tr ng-repeat="car in cars">
                            <td>{{car.carId}}</td>
                            <td>{{car.carName}}</td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </form>
</body>
</html>

```

.directive and .filter Service in AngularJS

Introduction

This explains the .directive and .filter service in AngularJS.

Angular provides many kinds of services, two of them are directive and filter. In this article I will show you an application in which both of the services are used.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the External JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript function, write this code in the head section:

```
<script>
    var x = angular.module('x', []);
    function ctrl($scope) {
        $scope.name = 'testing';
    }
    angular.module('mod', [])
        .filter('ifarray', function () {
            return function (input) {
                return $ isArray(input) ? input : [];
            }
        })
    
```

```

        }
    })
.directive('list', function ($compile) {
    return {
        restrict: 'E',
        terminal: true,
        scope: { val: 'evaluate' },
        link: function (scope, element, attrs) {
            if (angular.isArray(scope.val)) {
                element.append('<div>+ <div ng-
repeat="v in val"><list val="v"></list> </div></div>');
            } else {
                element.append(' - {{val}}');
            }
            $compile(element.contents())(scope.$new());
        }
    }
});
angular.module('x', ['mod']);

</script>

```

Here I created a module whose name is "x", this module will be called in the ng-app directive.

After this I created a function whose name is "ctrl", in this function one initial value is passed in the variable "name".

After creating the function I used the filter and directive services, in the directive service I passed restrict as "E", earlier in my previous articles I told you that restrict can be any of the three types "A, E and AE". "E" will match the element name only.

In this directive service I applied an "if" loop that will check whether or not the Array is used, if the array is used then it will append some controls to the element. I used a <list> control that is not a control, it's a customized control that is not available in HTML.

Step 3 - Now our work on JavaScript is completed and we can move to the design part of our application.

Write this code in the body section:

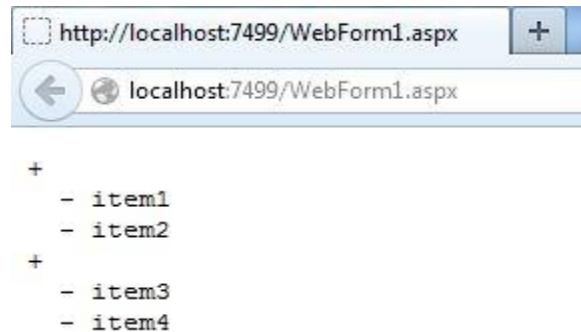
```
<body>
  <form ng-app="x" id="form1" runat="server">
    <pre>
      <list val="['item1','item2','[item3','item4']]">
        </list>
    </pre>
  </form>
</body>
```

Here I bound the module name with the form using the ng-app directive.

I used the custom element that was declared in the second step, in this <list> I bound some initial values that will be seen in the output window.

Now our application is created and is ready for execution.

Output: On running the application you will get an output like this:



You can see that the <list> is showing the output but it was not any official element.

The complete code of this application is as follows:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
  <script src="JavaScript1.js"></script>
```

```

<script>
    var x = angular.module('x', []);
    function ctrl($scope) {
        $scope.name = 'testing';
    }
    angular.module('mod', [])
        .filter('ifarray', function () {
            return function (input) {
                return $ isArray(input) ? input : [];
            }
        })
        .directive('list', function ($compile) {
            return {
                restrict: 'E',
                terminal: true,
                scope: { val: 'evaluate' },
                link: function (scope, element, attrs) {
                    if (angular.isArray(scope.val)) {
                        element.append('<div>+ <div ng-
repeat="v in val"><list val="v"></list> </div> </div>');
                    } else {
                        element.append(' - {{val}}');
                    }
                    $compile(element.contents())(scope.$new());
                }
            }
        });
    angular.module('x', ['mod']);
</script>
</head>
<body>
    <form ng-app="x" id="form1" runat="server">
        <pre>
            <list val="['item1','item2',['item3','item4']]">
                </list>
        </pre>
    </form>
</body>
</html>

```

\$httpBackend Service in AngularJS

Introduction

This introduces the \$httpBackend Service in AngularJS.

We can do two types of testing using AngularJS, the first is Unit Testing and the second is End to End testing, in other words E2E testing.

During Unit Testing Angular isolates the code and checks each part of code but in End to End all the code is checked by running the entire code.

Here I will show a small example of how E2E testing will work.

Step 1 - First of all you need to add two external files to your application, they are:

- angular.min.js
- angular-mocks.js

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script src="http://code.angularjs.org/1.0.3/angular-mocks.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will create a simple application where this service along with it's supportive services will be used.

First I will work on the JavaScript function. Write this code in the head section of your application:

```
<script>
var app = angular.module('mod', ['ngMockE2E']);

app.run(function($httpBackend) {
    $httpBackend.whenGET('/change').respond("hello");
});

app.controller('x', function($scope, $http) {
    $scope.doGet = function() {
        $http.get('/change').success(function(data) {
            $scope.clicked = data;
        });
    };
});
</script>
```

Here I created an object for the angular.module in which "mod" and "[ngMockE2E]" are defined. mod is a variable that will be used to bind the application using the ng-app. ngMockE2E shows that it will be an End to End Testing type.

After this \$httpBackend is defined that will check for getting the value from the user, if it does get something from the user then it will respond with a message such as "hello".

After this a controller is created in which the function "x" is defined, in this function a variable is used with service \$scope, this variable will hold the output provided through \$httpBackend.

Our work on JavaScript is completed and we can move to the design part.

Step 3 - Write this code in the body section:

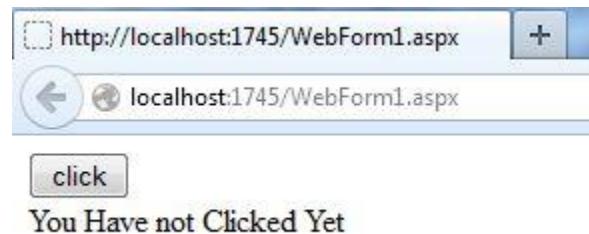
```
<form id="form1" runat="server">
    <div ng-controller="x">
        <input type="button" ng-click="doGet()" value="click"/>
        <div ng-bind="clicked || 'You Have not Clicked Yet'">?</div>
    </div>
</form>
```

Here I bound the first Div with the function "x" using the ng-controller.

In this Div I took a Button and a Label, the button click is bound to the doGet() function and the label is bound to the variable clicked.

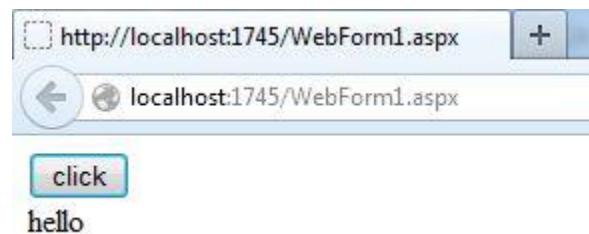
Now our application is created and can be executed.

Output: On running the application you will get an output like this one:



Here you can see that a button and that value is shown that was in the "or" section of Div.

Now if I click on the button then that will be shown that was provided in the default value at the JavaScript section as in the following:



The complete code of this application is as follows:

```
<html ng-app="mod" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script src="http://code.angularjs.org/1.0.3/angular-mocks.js"></script>
    <script>
        var app = angular.module('mod', ['ngMockE2E']);

        app.run(function($httpBackend) {
            $httpBackend.whenGET('/change').respond("hello");
        });

        app.controller('x', function($scope, $http) {
            $scope.doGet = function() {
                $http.get('/change').success(function(data) {
                    $scope.clicked = data;
                });
            };
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <input type="button" ng-click="doGet()" value="click"/>
            <div ng-bind="clicked || 'You Have not Clicked Yet'">?</div>
        </div>
    </form>
</body>
</html>
```

Sanitize Data Using AngularJS

Introduction

In this I will tell you how to sanitize data using AngularJS.

As you know, in normal life sanitize means to make things clean and in code we also need to make things clean and bug free, AngularJS provides a directive that can help us to eliminate harmful code, "ngSanitize".

ngSanitize doesn't allow the suspicious code to be executed so it can't attack our system, we can allow the code to bypass the sanitization at certain positions by applying some small conditions, here I created a small application that will help to understand how to apply Sanitize and also bypass the Sanitize.

Step 1 - First of all you need to add the following two external Angular.js files to your application:

1. angular.min.js
2. angular-sanitize.min.js

For this you can go to the AngularJS official site. After downloading the external files you need to add these files to the Head section of your application.

```
<head runat="server">
  <title></title>
  <script src="angular.min.js"></script>
  <script src="angular-sanitize.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript section of this application. Write this code in the head section of your application:

```
<script>
```

```
angular.module('htmldoc', ['ngSanitize']).controller('x', function ($scope, $sce) {
    $scope.html = '<p style="color:blue">Hey!! Come and ' +
        '<em style="color:Red" onmouseover="this.textContent=\'' + Click\'">Mouse Hover</em>\n' +
        'Over Me</p>';
    $scope.withoutSanitize = function () {
        return $sce.trustAsHtml($scope.html);
    };
});
</script>
```

Here I declared "htmldoc" using the angular.module, this "htmldoc" is the value that will be provided in the ng-app. Then 'ngSanitize' is declared that will be used to sanitize the code.

Then I provided the name of the controller as "x", then you can see that I provided an initial value in which many HTML tags are used along with a mouseover function.

After this I have created one more function named "withoutSanitize", in this function whatever we had declared in the variable "html" is assumed to be trusted code and not sanitized.

Our work on ViewModel is completed and now we just need to work on the View part or design part of this application.

Step 3 - Write this code in the Body tag:

```
<form id="form1" runat="server">
    <div ng-controller="x">
        <p ng-bind-html="html"></p>
        <div ng-bind-html="withoutSanitize()"></div>
    </div>
</form>
```

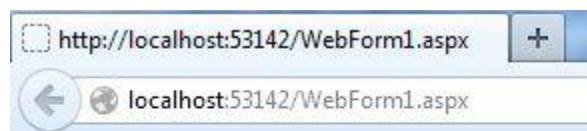
Here the first div is bound to the function x() using the ng-controller, in this div a <p> tag and a<div> tag is used.

The <p> tag is bound to the variable "html" using the ng-bind-html and the <div> tag is bound to withoutSanitize again using the ng-bind-html.

So, both tags are bound using the ng-bind-html but both are bound to various variables, the first one will allow the automatic sanitization but the second one will bypass the Sanitize.

Now our application is ready for execution.

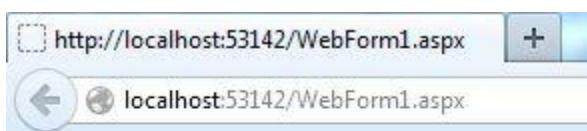
Output: On running the application you will get output like this:



Hey!! Come and *Mouse Hover* Over Me

Hey!! Come and *Mouse Hover* Over Me

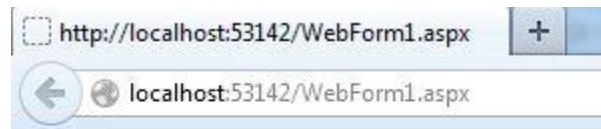
You can see that the first line of text is simply showing the data without any color added to that, mousehover is also not working on this first line of text as you can see here:



Hey!! Come and *Mouse[Hover* Over Me

Hey!! Come and *Mouse Hover* Over Me

But as I hover over the second line you can see that the text is change.



Hey!! Come and *Mouse Hover* Over Me

Hey!! Come and *Click* Over Me

The complete code of this application is as follows:

```

<html ng-app="htmldoc" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script src="angular-sanitize.min.js"></script>
    <script>
        angular.module('htmldoc', ['ngSanitize']).controller('x', function ($scope, $sce) {
            $scope.html = '<p style="color:blue">Hey!! Come and ' +
                '<em style="color:Red" onmouseover="this.textContent=\'' + Click + '\'>Mouse Hover</em>\n' +
                'Over Me</p>';
            $scope.withoutSanitize = function () {
                return $sce.trustAsHtml($scope.html);
            };
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <p ng-bind-html="html"></p>
            <div ng-bind-html="withoutSanitize()"></div>
        </div>
    </form>
</body>
</html>

```

\$cookies Service in AngularJS

Introduction

In this I will tell you about the \$cookies service in AngularJS.

AngularJS provides many type of directives and services that can be used for achieving various types of functionalities, AngularJS provides one more directive, in other words ngCookie and a service that's \$cookieStore, these directives and services can be used to store values into cookies.

Here I will show a simple example by which you will be able to understand how you can use the cookies feature of Angular.

Use the following procedure-

Step 1 - First of all you need to add the following two external Angular.js files to your application:

- angular.min.js
- cookies.js

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript section of this application.

Write this code in the head section:

```
<script>
    angular.module('app', ['ngCookies']);
    function x($scope, $cookies) {
        $scope.initially = "anubhav";
        $scope.setCookieValue = function () {
            $cookies.test = $scope.initially;
        }
        $scope.getCookieValue = function () {
            return $cookies.test;
        }
    }
</script>
```

Here I took an Angular module in which the "app" and "ngCookies" directives are defined; "app" is the name that will be bound with the ng-app directive.

I created a function "x" in which \$scope and \$cookies are declared.

After this I defined an initial value in the variable "initially" then I created a function that sets the value of Cookie, in this a variable "test" is declared that will receive the value from the variable "initially".

I created one more function to be used to get the value from Cookie. Now our work on JavaScript is completed and we can move to the design part.

Step 3 - Write this code in the body section:

```
<body>
    <form id="form1" runat="server">
        <div ng-app="app">
            <div ng-controller="x">
                Cookie Current Value: {{getCookieValue()}} <br/>
                <input type="text" ng-model="initially" />
                <button ng-click="setCookieValue()">Update cookie</button>
            </div>
        </div>
    </form>
</body>
```

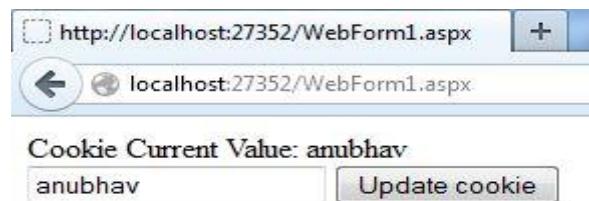
Here I bound the div using the ng-app, after this second div is bound with the function "x".

I had bound the "getCookieValue" in the div along with some text associated with it.

I also took a TextBox and a button, the TextBox is bound to the initial value of the variable "initially" and the button is bound to the "setCookieValue()" function.

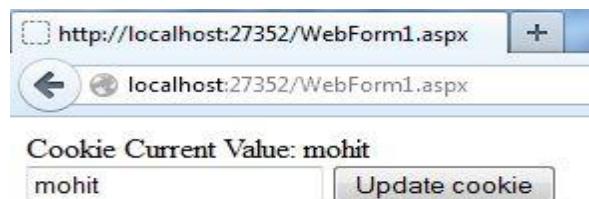
Now our application is ready for execution.

Output: On running the application you will get output like this:



Here Cookie and TextBox have similar values that were declared as the initial value in the JavaScript section.

Cookie will hold that value provided in the TextBox, so if I change the value in the TextBox and click on the button then the Cookie value will be updated.



Even if I run the application again without closing the browser then the output window will have the same value that was stored previously in the cookie, for example:



You can see that the TextBox is showing the default value that was provided earlier in the head section but the cookie has the same value that was provided in the previous tab.

The complete code of this application is as follows:

```

<html ng-app="app" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script src="//raw.github.com/angular/angular.js/master/src/ngCookies/cookies.js"></script>
    <script>
        angular.module('app', ['ngCookies']);

        function x($scope, $cookies) {
            $scope.initially = "anubhav";

            $scope.setCookieValue = function () {
                $cookies.test = $scope.initially;
            }

            $scope.getCookieValue = function () {
                return $cookies.test;
            }
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-app="app">
            <div ng-controller="x">
                Cookie Current Value: {{getCookieValue()}} <br/>
                <input type="text" ng-model="initially" />
                <button ng-click="setCookieValue()">Update cookie</button>
            </div>
        </div>
    </form>
</body>
</html>

```

Cookies in AngularJS

Introduction

In this I will tell you about Cookies in AngularJS.

AngularJS provides many types of directives and services for various types of functionalities, AngularJS provides one more directive, ngCookie and a service, \$cookieStore. These directives and services can be used to store some values in cookies.

Here I will show a simple example by which you will be able to understand how to use the cookies feature of Angular.

Use the following procedure-

Step 1 - First of all you need to add the following two external Angular.js files to your application:

- angular.min.js
- cookies.js

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file, the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript section of this application.

Write this code in the head section:

```
<script>
angular.module('app', ['ngCookies']);
function x($scope, $cookieStore) {
    $cookieStore.put('Name', 'Anubhav');
    $scope.cookie = $cookieStore.get('Name');
}
</script>
```

Here I took an Angular module in which the "app" and "ngCookies" directives are defined, "app" is the name that will be bound to the ng-app directive.

I created a function "x" in which \$scope and \$cookieStore are declared.

After this I have used the \$cookieStore to define some initial values and then called them in a variable named as "cookie".

Now our work on JavaScript is completed and we can move towards the Design part:

Step 3 - Write this code in the body section:

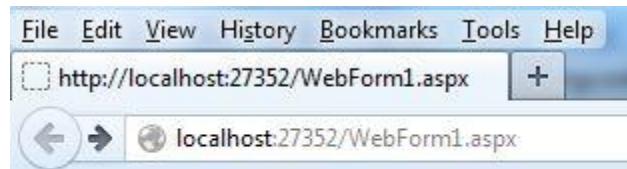
```
<body>
<form id="form1" runat="server">
<div ng-controller="x">
    Cookie Value: {{cookie}}
</div>
</form>
</body>
```

I had bound the div to the function "x" using the ng-controller.

I then bound the initial value declared in the head section using the {{cookie}}.

Now our application is created and is ready for execution.

Output: On running the application an output like this can be seen:



Cookie Value: Anubhav

You can see that the initial value is available in the output window.

The complete code of this application is as follows:

```
<html ng-app="app" xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script src="//raw.github.com/angular/angular.js/master/src/ngCookies/cookies.js"></script>
    <script>
        angular.module('app', ['ngCookies']);
        function x($scope, $cookieStore) {
            $cookieStore.put('Name', 'Anubhav');
            $scope.cookie = $cookieStore.get('Name');
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            Cookie Value: {{cookie}}
        </div>
    </form>
</body>
</html>
```

\$interval Service Provided by AngularJS

Introduction

In this I will tell you about the \$interval service provided by AngularJS.

An \$interval service can be used to trigger any function at a specific milliseconds of time interval, this will trigger the function repeatedly at a given interval of time.

Step 1 - First of all you need to add two external Angular.js files to your application, they are:

- angular.min.js
- cookies.js

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript section of this application.

Write the following code in the head section:

```
<script>
    var $interval = function (callback, interval) {
        (function timeout() {
            setTimeout(function () {
                callback();
                timeout();
            }, interval);
        })();
    };
</script>
```

```

        }, interval);
    })();
};

$interval(function () {
    document.body.innerHTML += 'I am Generated Through $interval <br>';
}, 1000);
</script>

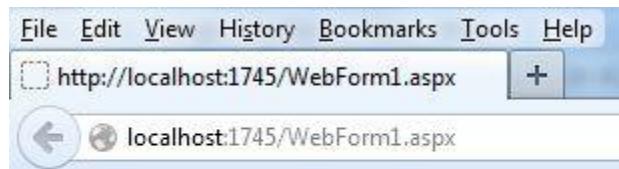
```

Here I created a function in which the variables callback and interval are defined.

I created a function in which some HTML text is provided, this HTML text will be printed repeatedly after 1000 ms.

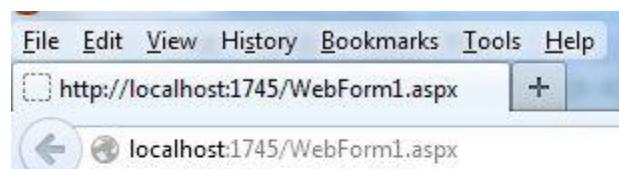
Now you don't need to provide anything in the body section. Just run the application and the output will be in front of you.

Output: On running the application you will get an output like this:



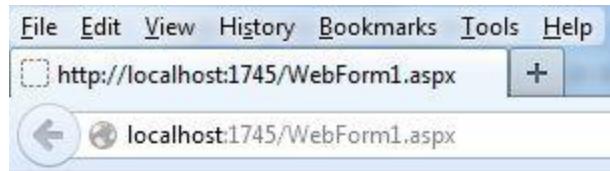
I am Generated Through \$interval

After a few ms some more data will be shown.



I am Generated Through \$interval
 I am Generated Through \$interval
 I am Generated Through \$interval
 I am Generated Through \$interval

Now some more:



I am Generated Through Sinterval
I am Generated Through Sinterval

NgInvalid in AngularJS

Introduction

In this I will tell you about "ngInvalid" in AngularJS.

ng-dirty in AngularJs helps to identify whether any changes are made by the user at run time or not. If you are making changes from the code section then it's Ok for ng-dirty and it will not provide any type of error but if you are making changes at run time then it will show an error or display the text according to the CSS defined in the ng-dirty class.

ng-invalid helps to identify whether some invalid data is entered by the user or can also help to identify whether the user has entered something or not.

Like ng-dirty, it is also defined in CSS as a class, then it is applied to the design section.

Now I will create a simple application and will show how to use ng-dirty in your application.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript section for this application.

Write this code in the head section:

```
<script>
  var mod = angular.module('mod', [])
  function x($scope) {
    $scope.UserEnter = 'Anu';
  }
</script>
```

Here I created a variable in which the angular. module is defined.

After this a function is created named "x", in this function a variable is used whose value is set to "Anu".

Step 3 - Now I will work on the CSS that will use the ng-invalid property of Angular.

Write this code just below the JavaScript that was provided in the step above.

```
<style>
  body { padding: 12px; }
  .ng-invalid { background-color: red; }
</style>
```

Here I apply the background-color property to the ng-invalid class, this means that the background color of that div will become red wherever ng-app is applied, if ng-app is applied in the head section then the entire background will become red.

Step 4 - Now I will work on the Body section, so write this code in the body:

```
<div ng-app>
  <form name="myForm" ng-controller="x">
    Please Enter Something:
    <input name="input" ng-model="UserEnter" required>
    <span class="error" ng-show="myForm.input.$error.required">Required</span><br>
    User Entered = {{UserEnter}}
    <br>
  </form>
</div>
```

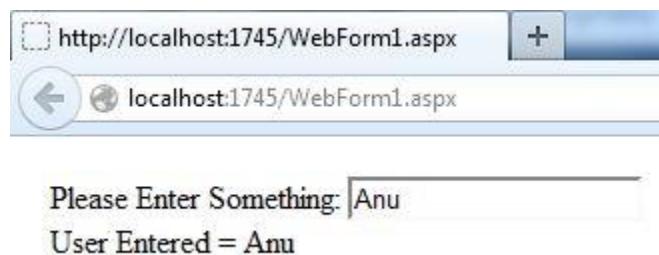
Here I created a container/parent Div bound to the ng-app, this means that on any mistake this div will become red and not the entire page.

In this div I created a TextBox bound using the ng-model, to this TextBox the required validation is also applied.

After these two spans are used, the first one will show an error message on finding the TextBox empty and the second one will show the text that was provided in the TextBox.

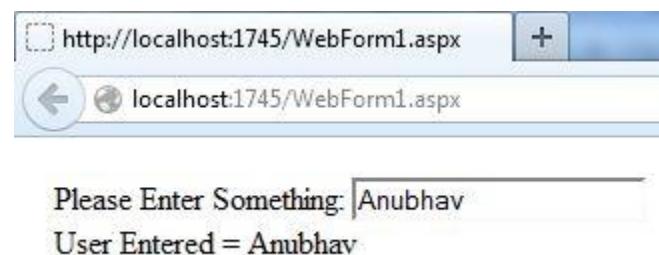
Now our application is created and is ready for it's execution.

Output: On running the application you will get output like this:

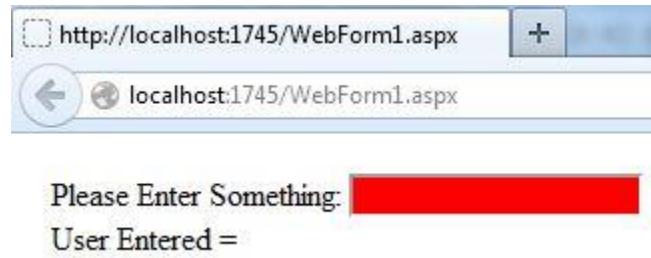


Here you can see that the default value is shown in both the TextBox and in front of the label.

Now if I change the value in the TextBox then similar changes will be seen in the label also:



Now if I remove the text from the TextBox then the red color will be applied to the TextBox.



The complete code of this application is as follows:

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        function x($scope) {
            $scope.UserEnter = 'Anu';
        }
    </script>
    <style>
        body { padding: 12px; }
        .ng-invalid { background-color: red; }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-app>
            <form name="myForm" ng-controller="x">
                Please Enter Something:
                <input name="input" ng-model="UserEnter" required>
                <span class="error" ng-show="myForm.input.$error.required">Required</span><br>
                User Entered = {{UserEnter}}
                <br>
            </form>
        </div>
    </form>
</body>
</html>

```

NgDirty in AngularJS

Introduction

In this I'm explains "ngDirty" in AngularJS.

ng-dirty in AngularJs helps to determine whether or not any changes have been made by the user at run time. If you are making changes from the code section then it's Ok for ng-dirty and it will not provide any type of error but if you are making changes at run time then it will show an error or display the text according to the CSS defined in the ng-dirty class.

ng-dirty is defined in the CSS as a class, then it is applied to the design section.

Now I will create a simple application and will show how you can use ng-dirty in your application.

Step 1 - First of all you need to add an external Angular.js file to your application, "angular.min.js".

For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the External JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript section for this application.

Write this code in the head section:

```
<script>
  var mod = angular.module('mod', []);

  function x($scope) {
    $scope.autoChangeValue = 1;
    setTimeout(function () {
      $scope.autoChangeValue = 5;
      $scope.$apply();
    }, 1000);
  }
</script>
```

Here I create a variable in which angular. module is defined.

After this a function named "x" is created, in the function a variable is used whose value is set to "1" but in the second or in the child function it's value is set to "5". So, two values are defined for the same variable but in two different functions.

Wherever these values will be bound, these values will be change after 1000 ms.

Step 3 - Now I will work on the Body section, so write this code in the body:

```
<form id="form1" runat="server">
  <p>If you make changes in the Textbox then background will change to Red</p>
  <div ng-controller="x">
    <div id="form">
      <input type="text" ng-model="autoChangeValue"/>
    </div>
  </div>
</form>
```

Here I bound the first div to the function "x" using the ng-controller, in this div a TextBox is created bound to the "autoChangeValue".

What "autoChangeValue" will do is, it will firstly show the first value, "1", and after 1000 ms this value will be changed to "5" automatically. But if you try to make changes by yourself then the background will be changed to "red" as specified in the JavaScript section.

Now our application is completed and can be executed.

Output: On running the application you will get an output like this:



If you make changes in the Textbox then background will change to Red

As I said in the beginning of this article, "1" will be shown initially but after 1000 ms it will be changed to "5".



If you make changes in the Textbox then background will change to Red

But if you try to make changes in the textboxes manually then the red color will be applied to the background.

The complete code of this application is as follows:

```
<html ng-app="mod" xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <%--<script src="http://code.angularjs.org/1.0.3/angular-mocks.js"></script>--%>
    <script>
        var mod = angular.module('mod', []);
        function x($scope) {
            $scope.autoChangeValue = 1;
            setTimeout(function () {
                $scope.autoChangeValue = 5;
                $scope.$apply();
            }, 1000);
        }
    </script>
    <style>
        body { padding: 12px; }
        .ng-dirty { background-color: red; }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <p>If you make changes in the Textbox then background will change to Red</p>
        <div ng-controller="x">
            <div id="form">
                <input type="text" ng-model="autoChangeValue"/>
            </div>
        </div>
    </form>
</body>
</html>
```

Ternary Operators in AngularJS

Introduction

In this I will tell you how to use Ternary Operators in AngularJS.

As the name suggests, the Ternary Operator contains three expressions. The Ternary Operator works like an if and else condition, if it finds the condition to be true then the first expression will be executed otherwise the second will be executed.

It has **two main operators**, in other words "?" (question) and ":" (colon). The "?" checks whether or not **the condition is true** and the ":" **separates the expressions** to be executed.

It's syntax is as in the following manner:

condition ? first_expression : second_expression;

Now I will create an example that will help you understand this syntax clearly.

Step 1 - First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".

For this you can go to the AngularJS official Site. After downloading the external file you need to add this file to the Head section of your application.

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
</head>
```

Step 2 - Now after adding the external JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

```
<html ng-app xmlns="http://www.w3.org/1999/xhtml">
```

Now I will work on the JavaScript of this application.

Write this code in the head section:

```
<script>
  var app = angular.module('app', []);
  function x($scope) {
    $scope.TernaryInAngular = false;
    $scope.checkConditions = function (value) {
      $scope.TernaryInAngular = !$scope.TernaryInAngular;
    };
  }
</script>
```

Here I created a variable in which the angular. module is defined.

After this a function is created named as "x", in this function a variable is declared named "TernaryInAngular", this variable is made Boolean and it is set to false initially.

After this another function is created to check the condition whether to make the variable active or inactive.

Now our work on JavaScript is completed and we can move towards the design section of this application.

Step 3 - Write this code in the body section:

```
<form id="form1" runat="server">
  <div ng-controller="x">
    <label ng-
class=" TernaryInAngular ? 'class1' : 'class2' ">Hello Everyone, You can see the changes in me</label>
    <br />
    <br />
    <input type="button" value="Chnage Class" ng-click="checkConditions()" ng-
class=" TernaryInAngular ? 'class1' : 'class2' "></input>
  </div>
</form>
```

Here the first div is bound to the function "x" using the ng-controller.

In this div a button and one label are taken, in both of these ng-class is used to bind them with two CSS classes, in other words "class1 and class2", these are bounded using the Ternary Operator. If the condition is found to be true then the first expression will be executed else the second expression will be executed.

The Click event of the button is bound to the function "checkCondition" that will change the class.

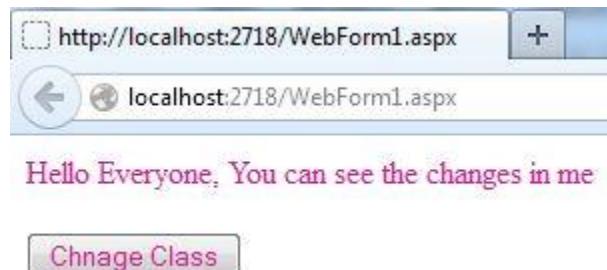
Step 4 - Write the following code in the head section to define the two classes, in other words "class1 and class2":

```
<style>
    .class1 { color: rgb(16, 136, 16); }
    .class2 { color: rgb(216, 27, 140); }
</style>
```

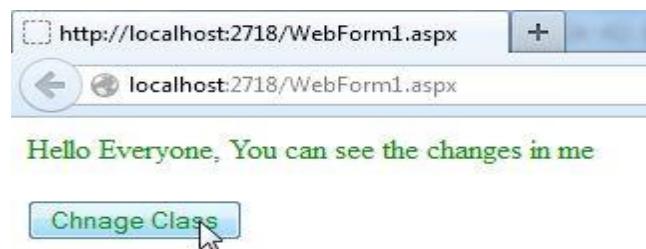
I changed the color in both of these classes, so a click of the button will change the color of the Label and of the Text written on the Button.

Now our application is created and is ready for it's execution.

Output: On running the application you will get output like this:



You can see that right now the color of the text is something like Pink, but as I click on the button this happens:



You can see that Ternary is working properly.

The complete code of this application is as follows:

```

<html ng-app xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script>
        var app = angular.module('app', []);

        function x($scope) {
            $scope.TernaryInAngular = false;

            $scope.checkConditions = function (value) {
                $scope.TernaryInAngular = !$scope.TernaryInAngular;
            };
        }
    </script>
    <style>
        .class1 { color: rgb(16, 136, 16); }
        .class2 { color: rgb(216, 27, 140); }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div ng-controller="x">
            <label ng-
class=" TernaryInAngular ? 'class1' : 'class2' ">Hello Everyone, You can see the changes in me </label>
            >
                <br />
                <br />
                <input type="button" value="Chnage Class" ng-click="checkConditions()" ng-
class=" TernaryInAngular ? 'class1' : 'class2' "></input>
            </div>
        </form>
    </body>
</html>

```

Make AJAX Call and Return JSON Using AngularJS

Introduction

This explains **How to get data in JSON format using AngularJS**.

Actually I was trying to make an AJAX call to a WebService using AngularJS. For that I tried this code:

```
<script>
  var myapp = angular.module('myApp', []);
  myapp.controller('control', function ($scope, $http) {
    var call = "WebService1.asmx/HelloWorld";
    $http.post(call).success(function (response) {
      $scope.value = response;
    })
    .error(function (error) {
      alert(error);
    });
  });
</script>
```

At the WebService I created a method that was returning a string; here I am just using Hello World that is provided in each WebService by default.

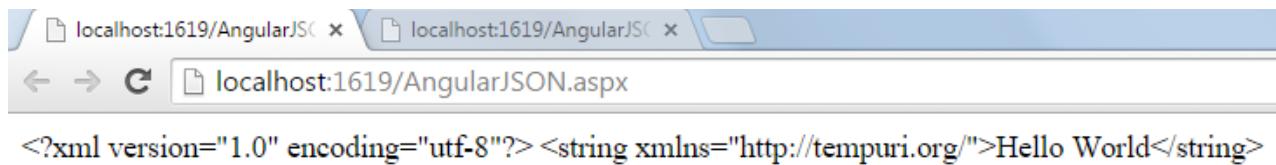
```
[System.Web.Script.Services.ScriptService]
public class WebService1: System.Web.Services.WebService
{
  [WebMethod]
  public string HelloWorld()
  {
    return "Hello World";
  }
}
```

After making the call I bound the value into a span using ng-bind.

```
<body>
  <form id="form1" runat="server">
```

```
<div ng-app="myApp" ng-controller="control">
    <span ng-bind="value"></span>
</div>
</form>
</body>
```

On running the code I got output like this.



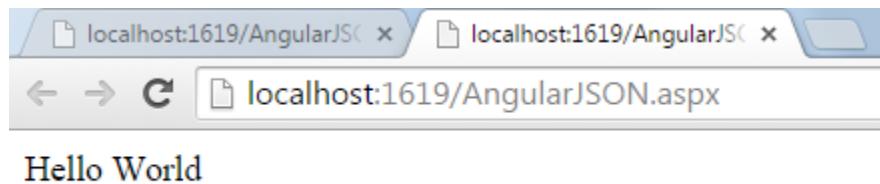
This was not the output I was expecting, I realized that the data is not coming in JSON format, so I had two possible solutions, either return the data from the WebService after converting it to JSON or make every incoming data JSON using AngularJS, I chose the second and changed the code to be as in the following:

```
<script>
var myapp = angular.module('myApp', []);
myapp.controller('control', function ($scope, $http) {
    $http({
        url: "WebService1.asmx/HelloWorld",
        dataType: 'json',
        method: 'POST',
        data: '',
        headers: {
            "Content-Type": "application/json"
        }
    }).success(function (response) {
        $scope.value = response.d;
    })
    .error(function (error) {
        alert(error);
    });
});
</script>
```

Now, you can see that the call is looking something like an AJAX call made using jQuery but it's from AngularJS, so some of you might feel that this way of calling is easier than the previous one but It's just the extended version of the previous call.

Here I have added the success and error functions so that I could determine what type of error I am getting if the code is not working.

Now, we can run the code and see the output:



Now, we are getting the expected result.

Call \$scope, Model of Child Page on Parent Page in AngularJS

Introduction

In one of my projects I encountered the situation where I needed to use the model value of one page on another page and these pages were in a parent-child relationship, in other words one page was opening in another one. At that time I came to understand that if we are in such a situation, we can call the model of the child on the parent page. This can be done in various ways, but here I will show the simplest one. In other words using the \$parent directive of Angular. Let's understand this using a simple example.

Step 1 - First of all I created a parent page. On this page I just created a div that is used to call the controller. Under this div I used a h1 tag, a button and one more div that will be used to include our child page. So, the code is something as in the following:

```
<!DOCTYPE html>
<html>

<head>
<script src="https://code.angularjs.org/1.3.0-beta.5/angular.js"></script>
<link href="style.css" rel="stylesheet" />
<script src="script.js"></script>
</head>

<body ng-app>
<div ng-controller="TestCtrl">
<h1>{{value}}</h1>
<div ng-include=""test.html""></div>
<button ng-click="fetch()">Go!!</button>
</div>
</body>
</html>
```

Step 2 - After this I created a child page and on this page a simple TextBox is used whose value will be used in the model and this model will be used in the parent page.

But this model is different from other models, in other words because the \$parent is written with it, that will allow the parent controller to access it. So, its scope is now extended to its parent. So only a single line of code is written on this page.

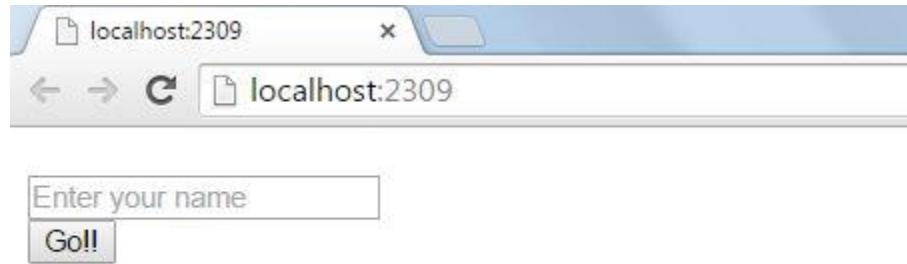
```
<input type="text" placeholder="Enter your name" ng-model="$parent.value">
```

Step 3: After this I worked on the .js file that has the following code:

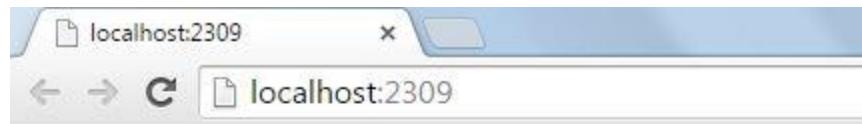
```
var TestCtrl = function($scope){  
    $scope.fetch = function(){  
        alert($scope.value);  
    };  
};
```

Here a controller is created whose name is "TestCtrl". In this controller two \$scopes are defined, the first one is the model that is called on the button click that was available on the parent page and the second \$scope has the model value that was provided at the child page. The second \$scope value will be alerted that will confirm that our code is working properly.

Output:

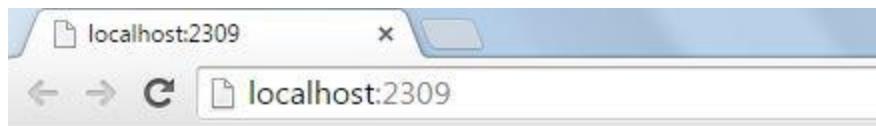


On running the application you can see that I got a TextBox and a button, the TextBox was available on the child page and the button was on the parent page.

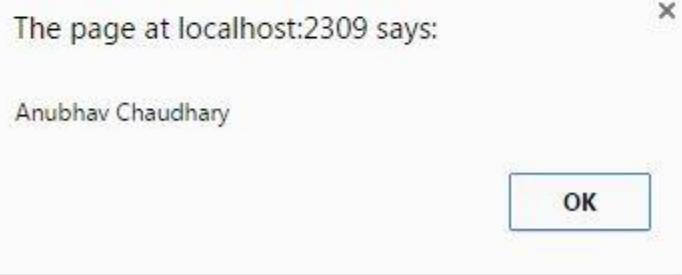


Anubhav Chaudhary

After this I wrote my name and as you can see the same name is provided by the h1 tag also, which means that our AngularJs is working properly.



Anubhav Chaudhary



As I click on the button, the same name can be seen in the alert box as well

Directive to Allow Decimal Numbers With a Single Decimal Point (.) Using AngularJS

Introduction

You often need to restrict the user to enter decimal numbers only but when you provide this functionality a major problem is encountered, the user can enter a decimal point (.) more than once in the TextBox and you encounter a medium bug. This article will provide the complete code to restrict users to decimal numbers with only a single decimal point allowed.

Step 1 - First of all you need to add AngularJs and jQuery to your page, just like here:

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script src="jquery-1.11.1.min.js"></script>
</head>
```

Step 2 - Now I am just adding a TextBox where I need to allow decimal numbers using a directive of AngularJs.

```
<body>
    <form ng-app="app" id="form1" runat="server">
        <div>
            <h3> Demo to Allow Decimal Numbers </h3>
            <hr />
            Provide Any Decimal Number: <input type="text"
                name="AntiPer"
                class="form-control"
                allow-decimal-numbers />
            <hr />
        </div>
    </form>
</body>
```

Step 3 - Now we need to work on the Angular code, the main part of this article.

```

<script>
  var app = angular.module('app', []);
  app.directive('allowDecimalNumbers', function () {
    return {
      restrict: 'A',
      link: function (scope, elm, attrs, ctrl) {
        elm.on('keydown', function (event) {
          var $input = $(this);
          var value = $input.val();
          value = value.replace(/\[^0-9\]/g, "")
          $input.val(value);
          if (event.which == 64 || event.which == 16) {
            // numbers
            return false;
          } if ([8, 13, 27, 37, 38, 39, 40, 110].indexOf(event.which) > -1) {
            // backspace, enter, escape, arrows
            return true;
          } else if (event.which >= 48 && event.which <= 57) {
            // numbers
            return true;
          } else if (event.which >= 96 && event.which <= 105) {
            // numpad number
            return true;
          } else if ([46, 110, 190].indexOf(event.which) > -1) {
            // dot and numpad dot
            return true;
          } else {
            event.preventDefault();
            return false;
          }
        });
      }
    });
  });
}

```

Here first of all I created a module with the name "app", the module creation is a necessary part if you want to use Angular. This module is added to the HTML page using the ng-app directive.

After this I created the directive with the name "allowDecimalNumbers". If you are creating any kind of directive using AngularJs then you must always check that name of that directive should be provided using '-' at your HTML page and these dashes (" - ") will be replaced by capital letters in the Angular code.

Since I had applied this directive on the attribute of the TextBox I had restricted this directive using "A", in other words Attribute.

I had applied this directive on the keydown event of the textbox.

These various conditions are applied to the ASCII values of the keys to allow or to stop them. You can see that a condition is added at the end that is allowing (46,110 and 190), this condition is used to allow the decimal point in the TextBox. In any case our ASCII values are not allowed by event.preventDefault() since it will remove that character and will revert back to the previous value.

But still the user is not restricted from entering the decimal point more than once. So, to remove this problem a small code is needed to be appending in this directive as in the following:

```
var findsDot = new RegExp(/\./g)
var containsDot = value.match(findsDot)
if (containsDot != null && ([46, 110, 190].indexOf(event.which) > -1)) {
    event.preventDefault();
    return false;
}
```

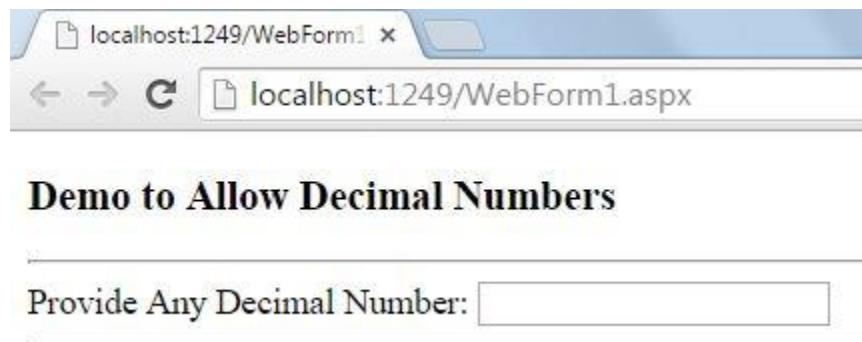
Here I am determining if our TextBox already has a decimal point, if it already has a decimal point and another character is entered with an ASCII value that matches the ASCII value of a decimal point then event.preventDefault() will remove this new decimal point.

So now our directive looks as in:

```
app.directive('allowDecimalNumbers', function () {
    return {
        restrict: 'A',
        link: function (scope, elm, attrs, ctrl) {
            elm.on('keydown', function (event) {
                var $input = $(this);
                var value = $input.val();
                value = value.replace(/\^0-9\./g, '')
                var findsDot = new RegExp(/\./g)
                var containsDot = value.match(findsDot)
                if (containsDot != null && ([46, 110, 190].indexOf(event.which) > -1)) {
                    event.preventDefault();
                    return false;
                }
            })
        }
    }
})
```

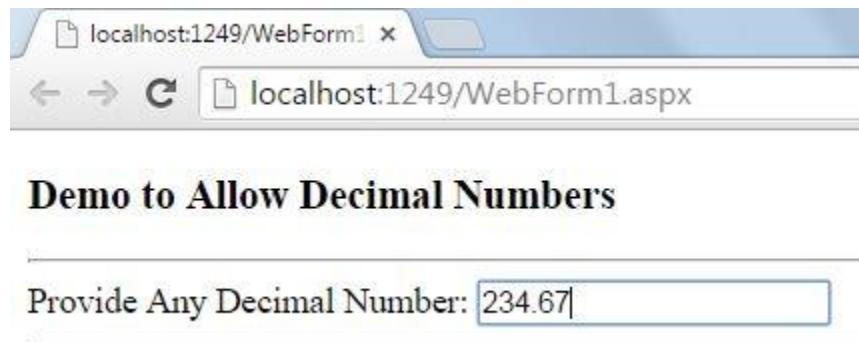
```
        }
        $input.val(value);
        if (event.which == 64 || event.which == 16) {
            // numbers
            return false;
        } if ([8, 13, 27, 37, 38, 39, 40, 110].indexOf(event.which) > -1) {
            // backspace, enter, escape, arrows
            return true;
        } else if (event.which >= 48 && event.which <= 57) {
            // numbers
            return true;
        } else if (event.which >= 96 && event.which <= 105) {
            // numpad number
            return true;
        } else if ([46, 110, 190].indexOf(event.which) > -1) {
            // dot and numpad dot
            return true;
        } else {
            event.preventDefault();
            return false;
        }
    });
}
});
```

Output: Now our application is created and we can check the output.



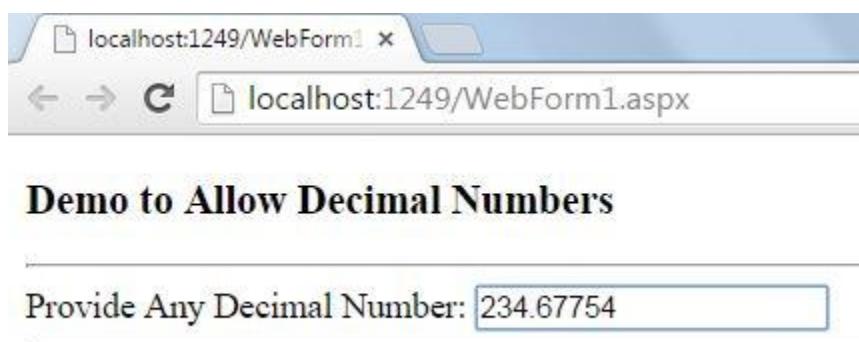
On running the application, a simple TextBox will be shown where decimal numbers are allowed.

If you press the numbers from anywhere in the key, they will be allowed but all the other characters except the decimal point will not be allowed to exist, even if they are special characters.



The screenshot shows a Microsoft Internet Explorer window with the URL `localhost:1249/WebForm1.aspx`. The page title is "Demo to Allow Decimal Numbers". Below the title is a text input field containing the value "234.67".

But if you try to enter more than one decimal point at any position in the TextBox then that decimal point will also not be allowed and will be removed.



The screenshot shows a Microsoft Internet Explorer window with the URL `localhost:1249/WebForm1.aspx`. The page title is "Demo to Allow Decimal Numbers". Below the title is a text input field containing the value "234.67754".

Directive to Allow Only Numbers Using AngularJs

Introduction

Before reading this article there is only one prerequisite condition; you should know how to create a directive in AngularJs and why we need to create them.

Step 1 - First of all you need to add AngularJs and jQuery to your page, just like here:

```
<head runat="server">
    <title></title>
    <script src="angular.min.js"></script>
    <script src="jquery-1.11.1.min.js"></script>
</head>
```

Step 2 - Now I am just adding a textbox where I need to allow only numbers using an AngularJs directive.

```
<body>
    <form ng-app="app" id="form1" runat="server">
        <div>
            <h3> Demo to Allow Numbers Only </h3>

            <hr />
            Provide Your Mobile Number: <input type="text"
                name="MobileNumber"
                class="form-control"
                allow-only-numbers />
            <hr />
        </div>
    </form>
</body>
```

Step 3 - Now we need to work on the Angular code, the main part of this article.

```
<script>
    var app = angular.module('app', []);
    app.directive('allowOnlyNumbers', function () {
        return {
            restrict: 'A',
            link: function (scope, elm, attrs, ctrl) {
```

```

elm.on('keydown', function (event) {
    if (event.which == 64 || event.which == 16) {
        // to allow numbers
        return false;
    } else if (event.which >= 48 && event.which <= 57) {
        // to allow numbers
        return true;
    } else if (event.which >= 96 && event.which <= 105) {
        // to allow numpad number
        return true;
    } else if ([8, 13, 27, 37, 38, 39, 40].indexOf(event.which) > -1) {
        // to allow backspace, enter, escape, arrows
        return true;
    } else {
        event.preventDefault();
        // to stop others
        return false;
    }
});
}
});
});
</script>

```

Here I first created a module named "app". The module creation is a necessary part if you want to use Angular. This module is added to the HTML page using the ng-app directive.

Then I created the directive with the name "allowOnlyNumbers". If you are creating any kind of directive using AngularJs then you must always check that the name is provided using "-" in your HTML page and these dashes "-" will be replaced by capital letters in the Angular code.

Since I had applied this directive to the attribute of the textbox I had restricted this directive using "A" for Attribute.

I had applied this directive on the keydown event of the textbox.

After those various conditions are applied to the ASCII values of the keys to allow or to stop them. In any case our ASCII values are not allowed. event.preventDefault() will remove that character and will revert to the previous value.

But still I had the serious problem of all the characters that were entered by pressing the **shift + key** were not removed by this code, so I used simple jQuery at the start of the code to help prevent this situation.

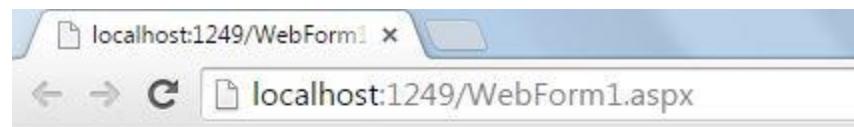
```
var $input = $(this);
var value = $input.val();
value = value.replace(/\[^0-9]/g, " ")
$input.val(value);
```

So now our directive looks as in:

```
app.directive('allowOnlyNumbers', function () {
  return {
    restrict: 'A',
    link: function (scope, elm, attrs, ctrl) {
      elm.on('keydown', function (event) {
        var $input = $(this);
        var value = $input.val();
        value = value.replace(/\[^0-9]/g, " ")
        $input.val(value);
        if (event.which == 64 || event.which == 16) {
          // to allow numbers
          return false;
        } else if (event.which >= 48 && event.which <= 57) {
          // to allow numbers
          return true;
        } else if (event.which >= 96 && event.which <= 105) {
          // to allow numpad number
          return true;
        } else if ([8, 13, 27, 37, 38, 39, 40].indexOf(event.which) > -1) {
          // to allow backspace, enter, escape, arrows
          return true;
        } else {
          event.preventDefault();
          // to stop others
          //alert("Sorry Only Numbers Allowed");
          return false;
        }
      });
    }
});
```

Output: Now our application is created and we can check the output.

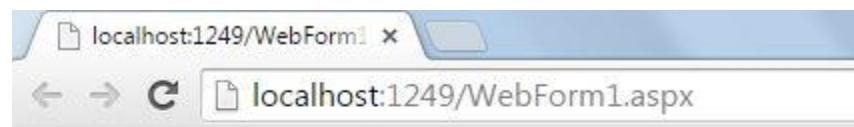
On running the application a simple textbox will be shown where only numbers are allowed.



Demo to Allow Numbers Only

Provide Your Mobile Number:

If you press the numbers from anywhere in the key, they will be allowed but all the other characters will not be allowed to exist, even if they are special characters.



Demo to Allow Numbers Only

Provide Your Mobile Number: 7867676679

Call Function, \$scope, Model of One Page From/on Second Page in AngularJS

Introduction

In previous I explained "How to Call \$scope, Model of Child Page on Parent Page in AngularJs". In that article I showed how to use "\$parent" to make the scope available at the parent page.

In that topic I also explained that their are other ways to call the scope value on other pages. This topic will show you one more way of using the scope value on another page and one more thing is included in this article that is you can also call the function created on another JavaScript page by some other page. This will be done using the **\$rootScope** of AngularJs.

AngularJs provides a very helpful feature that can be used to call \$scope values, functions of one page using some other page. This feature helps greatly in those cases where you want to access the \$scope value of various pages on one page or vice versa.

Let's create a sample project where I can elaborate upon both of these features.

Step 1 - First of all I created a HTML page. In this page I created a div to call the controller. Under this div I used an h1 tag, a button and one more div that will be used to include our child page, so the code is something as in the following:

```
<!DOCTYPE html>
<html>

<head>
<script src="https://code.angularjs.org/1.3.0-beta.5/angular.js"></script>
<link href="style.css" rel="stylesheet" />
<script src="script.js"></script>
<script src="testRootScript.js"></script>
</head>

<body ng-app>
<h3>
I am First Page
```

```

</h3>
<hr />
<div ng-controller="TestCtrl">
  <h1>{{value}}</h1>
  <div ng-include=""test.html""></div>
  <button ng-click="fetch()">Go!!</button>
</div>
</body>
</html>

```

Step 2 - After this I created another page and on this page again I created a div that is used to call another controller, so both pages have their own controllers. In this div a simple Text Box is used whose value will be used in the model and this model will be used in the first page.

```

<!DOCTYPE html>
<html>

<head>
  <script src="https://code.angularjs.org/1.3.0-beta.5/angular.js"></script>
  <link href="style.css" rel="stylesheet" />
  <script src="testRootScript.js"></script>
  <script src="script.js"></script>
</head>

<body ng-app>
  <hr />
  <h4>
    I am on Second Page
  </h4>
  <div ng-controller="TestRootCtrl">
    <input type="text" placeholder="Enter your name" ng-model="value">
  </div>
  <hr />
</body>
</html>

```

Step 3 - Now I am working on the Angular part of the first page.

Add a JavaScript page and on that page add the Angular code as in the following code snippet:

```
var TestCtrl = function ($scope, $rootScope) {
    $scope.value = $rootScope.value;
    $scope.fetch = function () {
        $rootScope.callFunction();
        $scope.value = $rootScope.value;
    };
};
```

Here I created the controller with the name "TestCtrl", you can see that along with \$scope, \$rootScope is also available. \$rootScope is the parent of all the scopes. All pages can access the \$rootScope value.

Here I have created an object named as value in which the \$rootScope value is passed. Also in the next function I had called a function that is also defined using \$rootScope. These functions and objects are not on this page but lies somewhere else and still I will be able to use it's value and call the function.

Step 4 - Now I am adding one more JavaScript page where the second controller will be defined and all the preceding rootscope values will also exist.

```
var TestRootCtrl = function ($scope, $rootScope) {
    $rootScope.callFunction = function () {
        alert("Function Called Successfully");
        $rootScope.value = "Anubhav Chaudhary";
    };
};
```

Here I have created the controller with the name "TestRootCtrl", this controller is used on the second page.

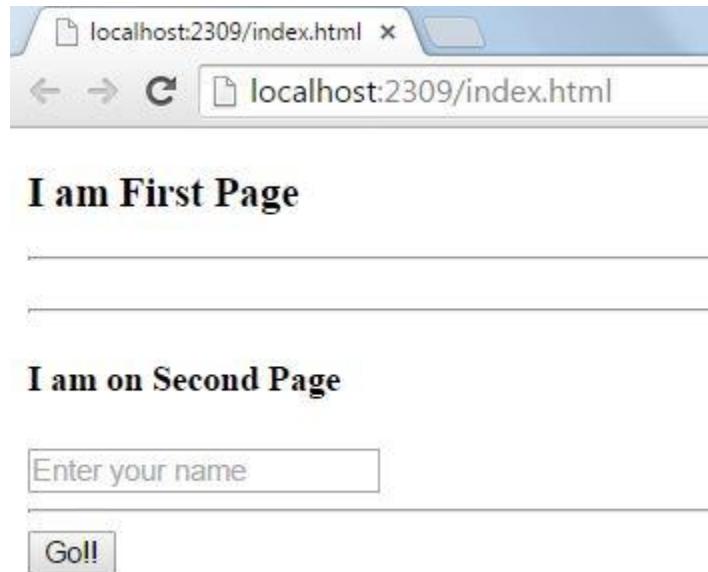
Here you can see that I created a function named as "callFunction", this function is defined using rootScope so it can be called from other pages.

Since the function will be called, we will get to understand since an alert message will be shown. We will also see the value of the rootScope object in all those places where we have bound it or passed it in some other scope object.

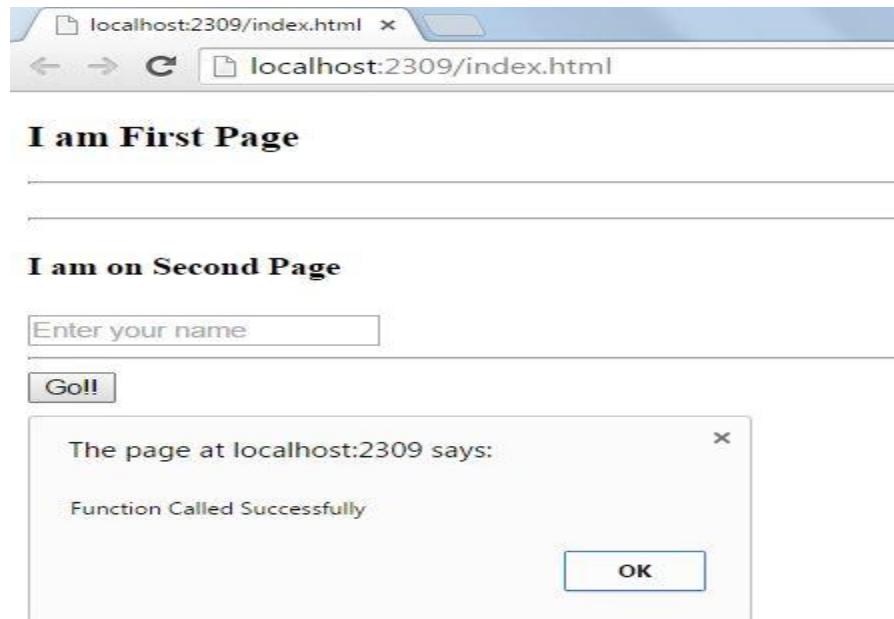
One major thing to keep in mind and to check is that the path of both of the script pages are passed on both the HTML pages otherwise your rootScope value will not be accessed and an error will be thrown.

```
<head>
<script src="https://code.angularjs.org/1.3.0-beta.5/angular.js"></script>
<link href="style.css" rel="stylesheet" />
<script src="script.js"></script>
<script src="testRootScript.js"></script>
</head>
```

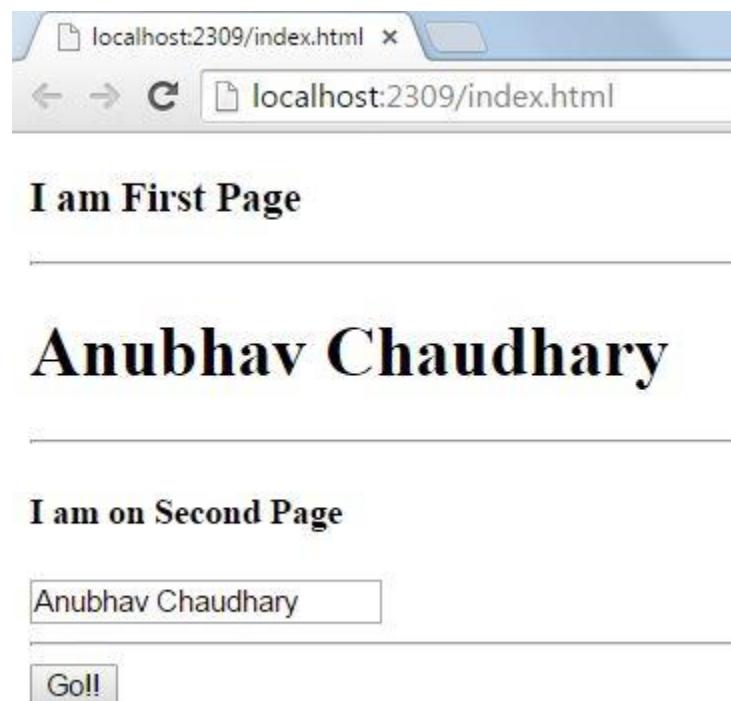
Output: On running the application you will see that the second page is coming along with the first page as it's child, since it was called using the ng-include directive on the first page.



As you click on the button first, an alert box will be shown that will confirm that our function is hit.



And as you close the window, the value of the rootScope object will pass into the control that was bound to this object.



Send Object of Objects From AngularJS to WebAPI

Introduction

In this I am explaining a really interesting and helpful thing, how to send an object of objects from AngularJs to WebAPI.

You might be making the calls from AngularJs to WebAPI or Web Service and you might encounter the situation where you need to send a large amount of data in the form of objects to your WebAPI. In that a case you can check this article and I am sure this will definitely help you.

Step 1 - First of all I am creating a demo form with some HTML input controls.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <script src="https://code.angularjs.org/1.3.0-beta.5/angular.js"></script>
    <script src="JavaScript1.js"></script>
</head>

<body ng-app>
    <hr />
    <h3>Send Objects of Objects Using AngularJS
    </h3>
    <hr />
    <div ng-controller="TestCtrl">
        <hr />
        <div>
            First Name:
            <input type="text" placeholder="Enter your name" ng-model="details.firstName">
            Last Name:
            <input type="text" placeholder="Enter your last name" ng-model="details.lastName">
        </div>
        <hr />
        <div>
            Mobile Number:
            <input type="text" placeholder="Enter your mobile number" ng-model="details.mobile">
            EmailId:
            <input type="text" placeholder="Enter your email id" ng-model="details.email">
        </div>
```

```

</div>
<input type="button" ng-click="SendData(details)" value="Submit" />
<hr />
<h4 style="color:blueviolet">{{value}}</h4>
</div>
<hr />
</body>
</html>

```

Here I created four textboxes, a button and a h3 tag that will confirm that our data has reached the API.

You can see that I had bound a model to each input control in the form of "details.firstName". A really good and helpful feature of Angular is that if it doesn't find any model specified on the HTML page in the JavaScript code then it first creates and then binds the value of it. I am using this feature of Angular. Here the details are working like a parent model in which multiple modules exist; I am using this approach because in this way I just need to pass the "detail" to the function on the click of a button and in that function all the values would be retrieved easily using this parent model. On the click of the button a function is called named SendData, in this function all the values are sent by just sending the parent model.

Step 2: Now we need to write our Angular Code, so for that add a JavaScript file and write code, something like the following:

```

var TestCtrl = function ($scope, $http) {
  $scope.SendData = function (Data) {
    var GetAll = new Object();
    GetAll.FirstName = Data.firstName;
    GetAll.SecondName = Data.lastName;
    GetAll.SecondGet = new Object();
    GetAll.SecondGet.Mobile = Data.mobile;
    GetAll.SecondGet.EmailId = Data.email;
    $http({
      url: "NewRoute/firstCall",
      dataType: 'json',
      method: 'POST',
      data: GetAll,
      headers: {
        "Content-Type": "application/json"
      }
    }).success(function (response) {

```

```
$scope.value = response;
})
.error(function (error) {
  alert(error);
});
} };
```

Here you can see that we have created the controller with the name "TestCtrl", in this controller apart from \$scope, \$http is also written, that's because \$http is the thing that makes the call in AngularJs.

In this controller I created a function; this is the same function that will be called on the click of the button. In the parameter the parent model is retrieved.

After this I created an object named "GetAll", this will work as the main object, in other words it will hold all other objects in it.

In this object I had added some properties that are getting the values from the "Data" parameter. Remember one thing, the naming convention for objects and properties should be proper because we will create some classes in the WebAPI that will have the same name as objects.

After providing some properties I had again created an object but this object will work as the property of the first object.

After this again some properties are created in this second object and they are also getting the values from our parameter.

After all this I created a \$http call that will call our WebAPI method. You can see that in the data I had simply provided this object and the wonderful thing is that the WebAPI will receive this object without any kind of problem.

In the success function I had passed the response to a scope that is bound to the h3 tag available at the bottom of our HTML page.

Step 3 - Final work is on the WebAPI, so add a WebAPI to your project and add code like this:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace WebApplication2.Controllers
{
    [RoutePrefix("NewRoute")]
    public class NewController : ApiController
    {
        public class GetAll
        {
            public string FirstName { get; set; }
            public string SecondName { get; set; }
            public SecondGet SecondGet;
        }

        public class SecondGet
        {
            public string Mobile { get; set; }
            public string EmailId { get; set; }
        }

        [Route("firstCall")]
        [HttpPost]
        public string firstCall(HttpRequestMessage request,
            [FromBody] GetAll getAll)
        {
            return "Data Reached";
        }
    }
}
```

Here first I created two classes and created some properties in these classes, you can see that I provided the same name to these classes that was provided when creating the objects on the JavaScript page.

The property name should also be same otherwise the data sent will not be mapped properly.

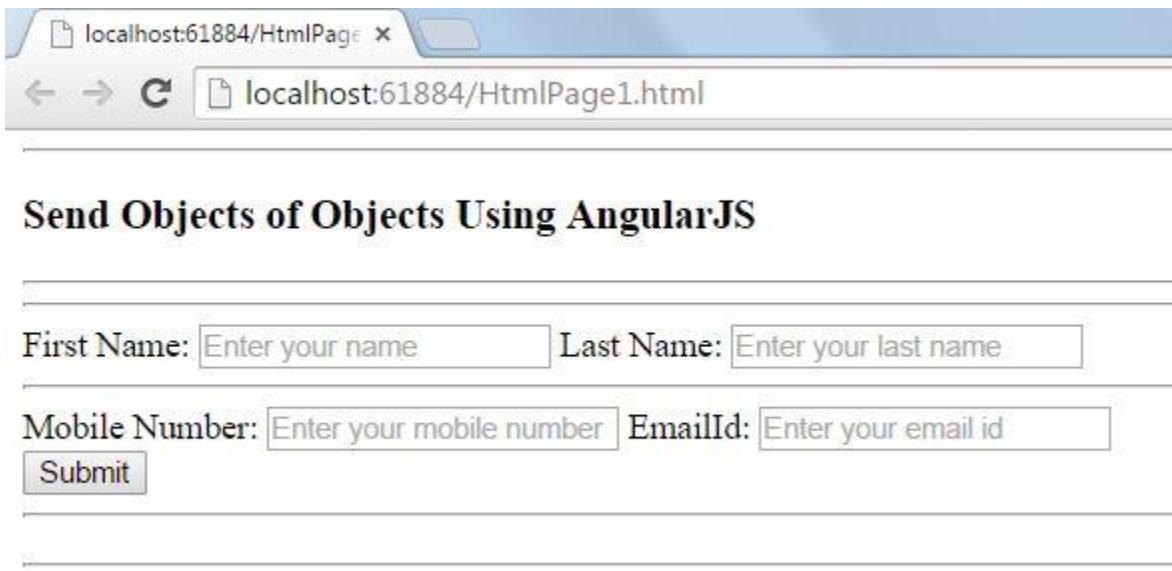
I used the second class as a property in the first class.

After creating the classes I created the `HttpPost` method and named it "firstCall". In this method I had `HttpRequestMessage` as the parameter because your method will not receive any kind of data from your Ajax call until and unless you allow your method by writing `HttpRequestMessage`. Since we are making the `HttpPost` call and we are sending a large amount of data from Angular we need to access that using the `[FromBody]`, if we would be passing any kind of small data like any integer or string then we can use `FromURI`.

Since the data will reach here, it will send some message in return and that message can be seen in the `h3` tag on the HTML page.

Now let's run the application and see the output.

Output: On running the application you will see a page with four fields and a button.



The screenshot shows a Microsoft Edge browser window. The address bar displays "localhost:61884/HtmlPage1.html". The main content area has a heading "Send Objects of Objects Using AngularJS". Below the heading are four input fields arranged in two rows: "First Name: Enter your name" and "Last Name: Enter your last name" in the first row; "Mobile Number: Enter your mobile number" and "EmailId: Enter your email id" in the second row. A "Submit" button is located below the second row of inputs.

Fill in your data and click on the button, you can check the data in the console by putting a debugger in the JavaScript code.

[Profiles](#) [Resources](#) [Audits](#) [Console](#)

script1.js × angular.js

```
TestCtrl = function ($scope, $http) {

    $scope.SendData = function (Data) { Data = Object {firstName: "Anubhav", lastName: "Chaudhary", mobile: "9876543210"};
        var GetAll = new Object(); GetAll = Object {FirstName: "Anubhav", SecondName: "Chaudhary", SecondGet: Object};
        GetAll.FirstName = Data.firstName; Data = Object {firstName: "Anubhav", lastName: "Chaudhary", mobile: "9876543210"};
        GetAll.SecondName = Data.lastName;
        GetAll.SecondGet = new Object();
        GetAll.SecondGet.Mobile = Data.mobile; Data = Object {firstName: "Anubhav", lastName: "Chaudhary", mobile: "9876543210"};
        GetAll.SecondGet.EmailId = Data.email;
        $http({
            url: "NewRoute/firstCall",
            dataType: 'json',
            method: 'POST',
            data: GetAll, GetAll = Object {FirstName: "Anubhav", SecondName: "Chaudhary", SecondGet: Object}
            headers: {

```

Screenshot of the Chrome DevTools Console tab showing the expanded object structure of `GetAll`:

```
> GetAll
< Object {FirstName: "Anubhav", SecondName: "Chaudhary", SecondGet: Object} ⓘ
  FirstName: "Anubhav"
  ▼ SecondGet: Object
    EmailId: "anubhav@test.com"
    Mobile: "9000000000"
    ► __proto__: Object
    SecondName: "Chaudhary"
    ► __proto__: Object
  > |
```

Now if you add a debugger at the API also, then you can see that the data is assigned to the properties created in the classes.

```

namespace WebApplication2.Controllers
{
    [RoutePrefix("NewRoute")]
    public class NewController : ApiController
    {
        public class GetAll
        {
            public string FirstName { get; set; }
            public string SecondName { get; set; }
            public SecondGet SecondGet;
        }

        public class SecondGet
        {
            public string Mobile { get; set; }
            public string EmailId { get; set; }
        }

        [Route("firstCall")]
        [HttpPost]
        public string firstCall(HttpRequestMessage request,
            [FromBody] GetAll getAll)
        {
            return "Data Reached";
        }
    }
}

```

The screenshot shows a Visual Studio code editor with the above C# code. A tooltip is displayed over the `getAll` variable in the `firstCall` method, showing its type as `WebApplication2.Controllers.NewController.GetAll` and its properties: `FirstName` (value: "Anubhav"), `SecondGet` (type: `WebApplication2.Controllers.NewController.SecondGet`), `EmailId` (value: "anubhav@test.com"), and `Mobile` (value: "9000000000").

The confirmation message can be seen in the browser.



Send Objects of Objects Using AngularJS

First Name:	<input type="text" value="Anubhav"/>	Last Name:	<input type="text" value="Chaudhary"/>
Mobile Number:	<input type="text" value="9000000000"/>	EmailId:	<input type="text" value="anubhav@test.com"/>
<input type="button" value="Submit"/>			

"Data Reached"

Get Attributes of Element in AngularJS Just Like jQuery

Introduction

JQuery provides the very good feature that when you have created any kind of clickkeyup/keydown and so on event then you can find all the attributes of that element in that function using the "this" keyword. But jQuery doesn't provide such kind of functionality (as far as I know, it might be possible that I am wrong). So, I was trying to access the attributes of an element in the angular code. Then I came to understand that we can access them, but with some small functionality. So, here I am explaining that trick/functionality to get the attributes.

Step 1 - First of all I have created a HTML page that will help us to understand this article in a very simple way.

```
<body ng-app>
  <h3>
    Page to show how to get attributes of html element in Angular code just like jquery have.
  </h3>
  <hr />
  <div ng-controller="AngularController">
    <input type="text" id="firstText" name="firstName" ng-keyup="showAttributes($event)" />
  </div>
</body>
```

Here I bound the controller to a div, this controller is not yet created but I provided this name in the HTML and I'll provide it in the Angular code as well.

In this div I took a simple HTML element that is of input type and added some attributes to this element, like "name", "id" and so on.

On the keyup of this element I called the function named "showAttributes", in this function I am sending the \$event as parameter. This parameter will help us to access the attributes.

Step 2 - After this I created a JavaScript file where we will write the AngularJs code. On this JavaScript file I had code something like this:

```
var AngularController = function ($scope) {

    $scope.showAttributes = function (element) {
        var prefix = 'Mr.';
        if (!(element.currentTarget.value.match('^Mr.'))) {
            element.currentTarget.value = prefix + element.currentTarget.value;
        }
    };
}
```

Here I have created a controller with the name "AngularController".

In this controller I created a function named "showAttributes". That is the function to be called on the keyup. The parameter of this function will help in accessing the attributes.

In this function I have created a variable to which some value is assigned.

Now I will check whether our TextBox has this predefined value in it. If not then this predefined value will come in front of the text and will work as a prefix.

The attributes of the element can be found only under the currentTarget.

Step 3 - Now call this JavaScript file on your HTML page in the head section as in the following:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <script src="https://code.angularjs.org/1.3.0-beta.5/angular.js"></script>
    <script src="AngularController.js"></script>
    <title></title>
</head>
<body ng-app>
    <h3>
        Page to show how to get attributes of html element in Angular code just like jquery have.
    </h3>
    <hr />
    <div ng-controller="AngularController">
        <input type="text" id="firstText" name="firstName" ng-keyup="showAttributes($event)" />
    </div>
</body>
</html>
```

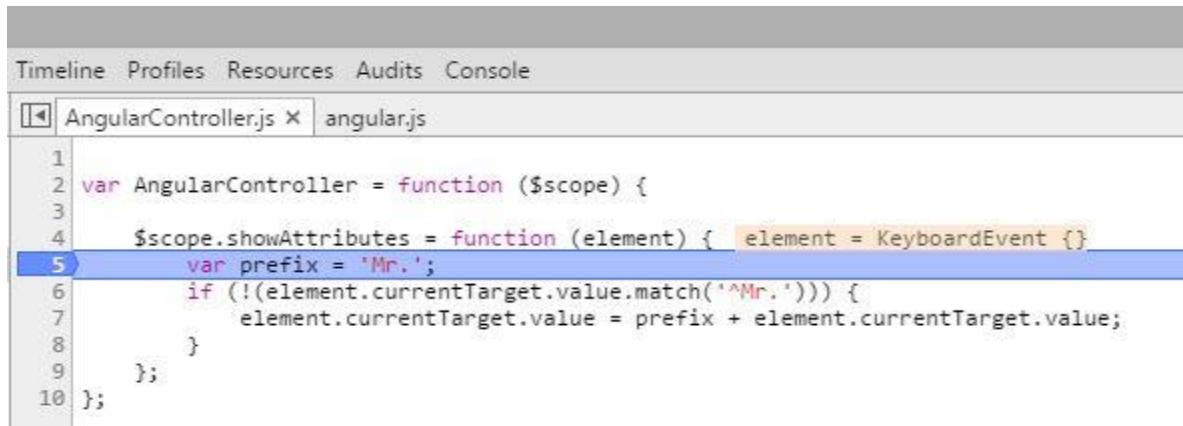
Output: Now we can run the code and can check whether our code is working properly or not.

On running the code, first of all a simple page will be seen.



The screenshot shows a browser window with the URL `localhost:63816/AngularPage.html`. The page title is "Page to show how to get attributes of html element in Angular code". Below the title is a single input text box.

But as you click on the TextBox and check the value in the console, you will find that the method is called perfectly and as you move forward you will see that all the attributes are available in the current target.



The screenshot shows the Chrome DevTools Console tab with the AngularController.js file open. The code is as follows:

```
1 var AngularController = function ($scope) {
2
3     $scope.showAttributes = function (element) { element = KeyboardEvent {}}
4         var prefix = 'Mr.';
5         if (!(element.currentTarget.value.match(/^\w+/))) {
6             element.currentTarget.value = prefix + element.currentTarget.value;
7         }
8     };
9
10 };
```

On the TextBox, **Mr.** will be append automatically.



The screenshot shows a browser window with the URL `localhost:63816/AngularPage.html`. The page title is "Page to show how to get attributes of html element in Angular". Below the title is a text box containing the value "Mr. Anubhav Chaudhary".

Set and Clear Timeout Using \$timeout in AngularJS

Introduction

In your projects you might want to call a particular method again and again after a particular interval of time, it can be used in those cases where we need to refresh a particular control so that new data can be seen. It can be done using the `$timeout` directive of AngularJS. But there is a problem with `$timeout` that even if you change the page it will keep on working and it might frustrate the user as page might refresh or it can effect the performance of page. In that case you need to stop the timeout, so, here I'll tell both the things i.e. How to start and stop the `$timeout` in AngularJS.

Step 1 - Here I am creating a simple example, you can make changes according to your requirement. First of all I added a HTML page where I will show the timer working and will provide a link to stop the timer as well (you can use this link to stop and to change the page as well).

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <script src="http://code.angularjs.org/angular-1.0.0rc11.min.js"></script>
  <script src="AngularController.js"></script>
  <title></title>
</head>
<body ng-app>
  <h3>
    Page to show how to set and clear timeout.
  </h3>
  <hr />
  <div ng-controller="AngularController">
    <h4>Let's Count: {{startCount}}</h4>
    <a ng-click="stopTimeout()">Click to destroy Timer</a>
  </div>
</body>
</html>
```

Here I had bind the div with controller named as "**AngularController**".

In this div I had taken a <h4> tag which is bind to that model which will be updated using the \$timeout directive.

After this I had provided the an anchor tag, on click of this anchor a function is called named as "stopTimeout()". This function will stop the timeout.

Step 2 - Now I will work on the Angular Code. So, for that I had taken a JavaScript page on which this code is written:

```
var AngularController = function ($scope, $timeout) {
    $scope.startCount = 0;
    $scope.startTimeout = function () {
        $scope.startCount = $scope.startCount + 1;
        mytimeout = $timeout($scope.startTimeout, 1000);
    }
    $scope.startTimeout();
    $scope.stopTimeout = function () {
        $timeout.cancel(mytimeout);
        alert("Timer Stopped");
    }
};
```

First of all I had created controller with same which was provided on the HTML page to bind the Div i.e. "AngularController". You can see that while creating this controller I had passed \$timeout along with \$scope, this is because if you don't pass this predefined directive then you will not be able to use it just like \$scope is not available when it is not passed while creating the controller and start giving the error.

In this controller I had taken a model named as "startCount" and set it's value to 0. After this I had created a function which will be called again and again. In this function I had incremented the startCount model by one. After this I had passed the \$timeout directive in a variable, passing it into a variable will help you to stop it.

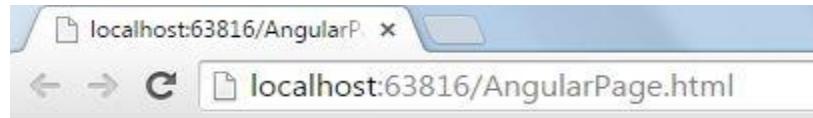
In \$timeout directive you need to pass two parameters, first one is the name of function which is to be called and other one is the interval of time in milliseconds after which this function is to be called. After creating the function I had called it from outside, this will work as kickoff.

Another function is created which will be used to stop the \$timeout. This is done by cancelling the variable in which timeout was called. An alert message will also be displayed to confirm that all things are working in right manner. If you want you're \$timeout directive to stop on page change then this code can be used:

```
$scope.$on("$destroy", function (event)
{
    alert("I am leaving");
    $timeout.cancel(mytimeout);
});
```

This code will work when controller is changing, \$destroy will destroy all those models which are passed in it.

Output: Now our coding part is done and we can run the application to see the result. On running the application, you will see that count has started and is incrementing on each second.



Page to show how to set and clear timeout.

Let's Count: 3

[Click to destroy Timer](#)

Now if we click on the anchor then our counter will stop and an alert message will be shown.



Page to show how to set and clear timeout.

Let's Count: 53

[Click to destroy Timer](#)



Show Data in Grid Format Using AngularJS and WEB API

Introduction

This explains how to show data in grid format using AngularJs and the Web API.

Step 1 - I first added some references for supporting AngularJs and BootStrap to this application as in the following:

```
<head>
  <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
  <script src="https://code.angularjs.org/1.3.0-beta.5/angular.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.0rc1/angular-route.min.js"></script>
  <script src="JavaScript1.js"></script>
  <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
</head>
```

Then I worked on the HTML code that will create the Grid.

```
<div ng-controller="TestCtrl" ng-init="firstCall()">
  <div class="table-responsive com-table">
    <table class="table table-bordered table-hover table-striped">
      <thead>
        <tr>
          <th width="5%">Customer ID</th>
          <th width="15%">Customer Name</th>
          <th width="15%">Email</th>
          <th width="20%">Mobile No.</th>
          <th width="25%">Address</th>
          <th width="20%">Registration Date</th>
        </tr>
      </thead>
      <tbody>
        <tr ng-repeat="CU in CustomerList">
          <td width="5%">{{CU.CustomerId}}</td>
          <td width="15%">{{CU.Name}}</td>
```

```

<td width="15%">{{CU.Email}}</td>
<td width="20%">{{CU.PhoneNumber}}</td>
<td width="25%">{{CU.Address}}</td>
<td width="20%">{{CU.Date}}</td>
</tr>
</tbody>
</table>

<div class="collapse" >
  <div class="well">
    </div>
</div>
</div>

```

In the first div you can see that I had called the controller and one more thing is written along with that, ng-init="" . What this ng-init does is it will call a specific function when initializing the controller for the first time. In simple words a function will be called automatically as soon as the page is loaded.

After that you can see that I had added a simple table of HTML, but this simple table is not so simple because ng-repeat will make this work as a grid. ng-repeat works like a foreach() for Angular, ng-repeat passes the data in an object and then we can use this object to bind specific properties to each li or tr tag.

That much is enough to create a simple grid.

Step 2 - After working on the HTML page it's time to move on to the Angular code. So for this I added a JavaScript page and on that page this code is written:

```

var TestCtrl = function ($scope, $http) {
  $scope.firstCall = function () {

    $http({
      url: "NewRoute/getDataForAngularGrid",
      dataType: 'json',
      method: 'GET',
      data: "",
      headers: {
        "Content-Type": "application/json"
      }
    })
  }
}

```

```
        }
    }).success(function (response) {
        debugger;
        $scope.CustomerList = response;
    })
    .error(function (error) {
        alert(error);
    });
}
}
```

You can see that first of all a controller is created and in that controller a single function is created, this is the same function that will be called on initialization.

In this function a call is made to the Web API method named "getDataForAngularGrid".

If everything is right, in other words if the call is made to the Web API, then the success function will be executed and in that success function I am sending data to a scope named "CustomerList". This is the same scope that is used to bind the values in the table.

Step 3 - After this I worked on the Web API.

Here I created a method whose name is "getDataForAngularGrid", in this method I simply made a call to the SQL Server and fetched my data from a specific table.

```
namespace WebApplication2.Controllers
{
    [RoutePrefix("NewRoute")]
    public class NewController : ApiController
    {
        public class GetAll
        {
            public int CustomerId { get; set; }
            public string Name { get; set; }
            public string Address { get; set; }
            public string Email { get; set; }
            public string PhoneNumber { get; set; }
            public string Date { get; set; }
        }
    }
}
```

```
[Route("getDataForAngularGrid")]
[HttpGet]
public List<GetAll> getDataForAngularGrid(HttpRequestMessage request)
{
    DataTable dt = new DataTable();
    List<GetAll> list = new List<GetAll>();
    try
    {
        SqlConnection con = new SqlConnection("YourConnection");
        con.Open();
        var query = "SP_getAllData";
        SqlCommand com = new SqlCommand(query, con); //creating SqlCommand object
        com.CommandType = CommandType.StoredProcedure;
        com.ExecuteNonQuery();
        con.Close();
        SqlDataAdapter adptr = new SqlDataAdapter(com);
        adptr.Fill(dt);
        for (int i = 0; i < dt.Rows.Count; i++)
        {
            GetAll GetAll = new GetAll();
            GetAll.CustomerId = Convert.ToInt32(dt.Rows[i]["customerid"]);
            GetAll.Name = dt.Rows[i]["name"].ToString();
            GetAll.Address = dt.Rows[i]["address"].ToString();
            GetAll.Email = dt.Rows[i]["email"].ToString();
            GetAll.PhoneNumber = dt.Rows[i]["phno"].ToString();
            GetAll.Date = dt.Rows[i]["date"].ToString();
            list.Add(GetAll);
        }
    }
    catch (Exception e)
    {
    }
    return list;
}
}
```

Here you can see that I had fetched the data and then I had sent this data back to the controller using a list of the class.

Now our coding part is completed and we can execute it to check the output.

Output: On running the application call will directly go to the Web API.

Web.config NewController.cs JavaScript1.js HtmlPage1.html

WebApplication2.Controllers.NewController

```

        con.Open();
        var query = "SP_getAllData";
        SqlCommand com = new SqlCommand(query, con); //creating SqlCommand object
        com.CommandType = CommandType.StoredProcedure;
        com.ExecuteNonQuery();
        con.Close();
        SqlDataAdapter adptr = new SqlDataAdapter(com);
        adptr.Fill(dt);
        for (int i = 0; i < dt.Rows.Count; i++)
    {
    }

```

DataSet Visualizer

customerid	name	address	email	phno	date
2079873637	Anubhav Chaudh...	rtetert	abc@abc.com	1234567890	03-10-2009 01:05...
1219337748	Mohit Chaudhary	hik	f@w.com	3453465487	04-05-2010 02:41...
1219337748	Swati Singh	refgerf	a@w.com	12345567788	04-05-2010 02:42...
1261585645	Sandeep Panghal	terter	tert@rest.dgdfg	1234567890	04-05-2010 23:21...

From here data will be sent to the Angular Success function.

Timeline Profiles Resources Audits Console

JavaScript1.js X angular.js

```

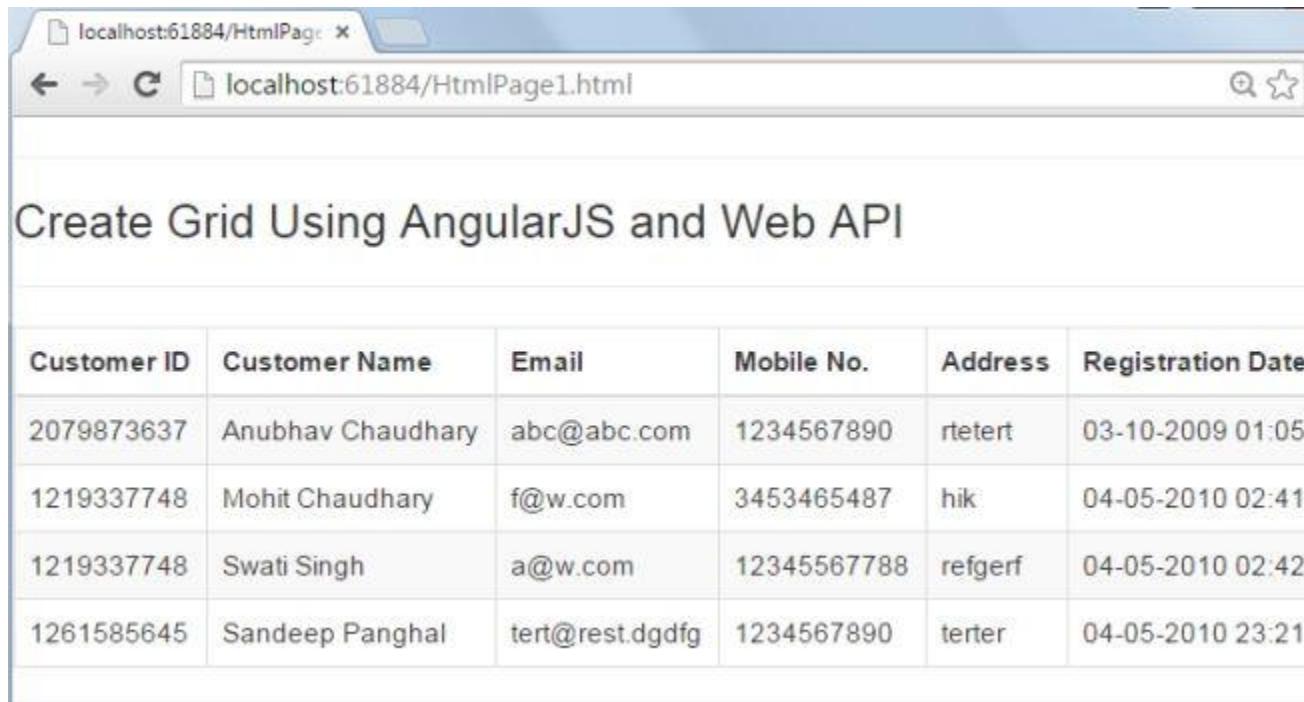
11
12     }).success(function (response) { response = [Object, Object, Object, Object]
13     debugger;
14     $scope.Customer
15     })
16     .error(function (error)
17     alert(error)
18   );
19
20   $scope.SendData = function()
21   var GetAll = new Ob
22   GetAll.FirstName =
23   GetAll.SecondName =
24   GetAll.SecondName =
25
26 } Line 13, Column 13

```

Array[4]

- ▶ 0: Object
- ▼ 1: Object
 - Address: "hik"
 - CustomerId: 1219337748
 - Date: "04-05-2010 02:41:45"
 - Email: "f@w.com"
 - Name: "Mohit Chaudhary"
 - PhoneNumber: "3453465487"
 - ▶ __proto__: Object
- ▶ 2: Object
- ▶ 3: Object

After this all the data will be shown in grid format.



The screenshot shows a web browser window with the URL `localhost:61884/HtmlPage1.html`. The page title is "Create Grid Using AngularJS and Web API". Below the title is a table with six columns: Customer ID, Customer Name, Email, Mobile No., Address, and Registration Date. The table contains five rows of data.

Customer ID	Customer Name	Email	Mobile No.	Address	Registration Date
2079873637	Anubhav Chaudhary	abc@abc.com	1234567890	rtetert	03-10-2009 01:05
1219337748	Mohit Chaudhary	f@w.com	3453465487	hik	04-05-2010 02:41
1219337748	Swati Singh	a@w.com	12345567788	refgerf	04-05-2010 02:42
1261585645	Sandeep Panghal	tert@rest.dgdfg	1234567890	terter	04-05-2010 23:21

Retain Focus On Invalid Field Using AngularJS Custom Directive

Introduction

In this I'll tell you about **How to Retain Focus on Invalid Field using AngularJS Custom Directive.**

Directives can be created for validation purpose but here I am not explaining those as some directives are already available for validation purpose, but you can download the source code and can get the directive from controller.js.

Let's see how to retain focus on invalid field through a simple application.

Step 1 - Firstly, I created an HTML page where some controls are created and angular validation is applied.

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
  <script src="Controller.js"></script>
</head>
<body>
  <form ng-app="TestFormMod" name="ApplicantCustomerProfile" ng-controller="TestFormCtrl" novalidate angular-validator >
    <div class="gloabl-form-content">
      <div class="row">
        <div class="col yellow-divider">
          <div class="col-first">
            <label><span class="asterisk">*</span>Name of father/Guardian</label>
          </div>
          <div class="col-second">
            <input name="guardian_or_father_name"
                  type="text"
                  name="fatherName"
                  ng-model="customer_profile.guardian_or_father_name"
                  validate-on="blur"
                  ng-pattern="/^[\ a-zA-Z]*$/"
```

```

        required
        required-message="'Please enter father name.'"
        invalid-message="'Only alphabets are allowed'"
        maxlength="100" />
    </div>
    <div class="col-third">
        <label>Name of Spouse</label>
    </div>
    <div class="col-four">
        <input name="spouse_name" type="text" ng-
model="customer_profile.spouse_name"
            validate-on="blur"
            ng-pattern="/^[ a-zA-Z]*$/"
            invalid-message="'Only alphabets are allowed'"
            maxlength="100" />
    </div>
</div>
<div class="row">
    <div class="col yellow-divider">
        <div class="col-third">
            <label><span class="asterisk">*</span>Phone Number</label>
        </div>
        <div class="col-four">
            <input type="text"
                name="annual_turnover"
                ng-model="customer_profile.annual_turnover"
                validate-on="blur"
                ng-pattern="/^([ 0-9.])*$/"
                required
                required-message="'Please enter mobile number'"
                invalid-message="'Only numbers are allowed.'" />
        </div>
    </div>
</div>
</div>
</div>
</form> </body> </html>
<style>
    input.apply-visited.ng-invalid:not(.apply-focus)
    {
        background-color: #ffeeee;
    }
</style>
```

Here you can see that I had applied different kind of validation on three controls: required, number only, alphabets only.

You can see in the form tag that I had applied a directive named "angular-validator", this is the same directive which is used to apply the validations and you can get that from source code.

Step 2 - Now let's create a JavaScript file where controller will be created.

I created a controller named "TestFormCtrl", here you can write any code according to your requirements.

```
angular.module('TestFormMod', [])

.controller('TestFormCtrl', function ($scope, $location, $rootScope) {
    //apply code here
})
```

Step 3 - After this I had created two directives which will help to retain focus on input fields.

```
.directive("parentDirective", function () {
    return {
        restrict: 'A',
        require: ['form'],
        controller: function () {
            // nothing here
        },
        link: function (scope, ele, attrs, controllers) {
            var formCtrl = controllers[0];
        }
    };
})

.directive('input', function () {
    return {
        restrict: 'E',
        priority: -1000,
        require: ['^?parentDirective', '^?angularValidator'],
        link: function (scope, elm, attr, ctrl) {
            if (!ctrl) {
                return;
            }

            elm.on('focus', function () {
```

```

elm.addClass('apply-focus');

scope.$apply(function () {
    ctrl.hasFocus = true;
});

elm.on('blur', function () {
    elm.removeClass('apply-focus');
    elm.addClass('apply-visited');

    scope.$apply(function () {
        ctrl.hasFocus = true;
        ctrl.hasVisited = true;

    });
});
});

}
};

});
);
});

```

First one is like a parent directive and second one will work only when we are having parent directive and validator directive. Second directive will help to retain focus on elements.

Now you just need to call this directive in the form tag of your page.

```
<form ng-app="TestFormMod" name="ApplicantCustomerProfile" ng-controller="TestFormCtrl" novalidate angular-validator parent-directive>
```

Here I am only providing code for those conditions where user will not be able to move from one control to second unless he / she fulfills the validations, but if user click on button which is on master page and have some other controller then this code is not enough to stop the user from navigation. In that case you need to do modifications in your code and need to check whether page is valid or not, if not then you just need to stop the navigation as this code will be still working.

Output: Let's run the application and see the output.

On running the application three textbox will appear.



*Name of father/Guardian

Name of Spouse

*Phone Number

Now just click on required textbox and try to leave the textbox without filling the value.

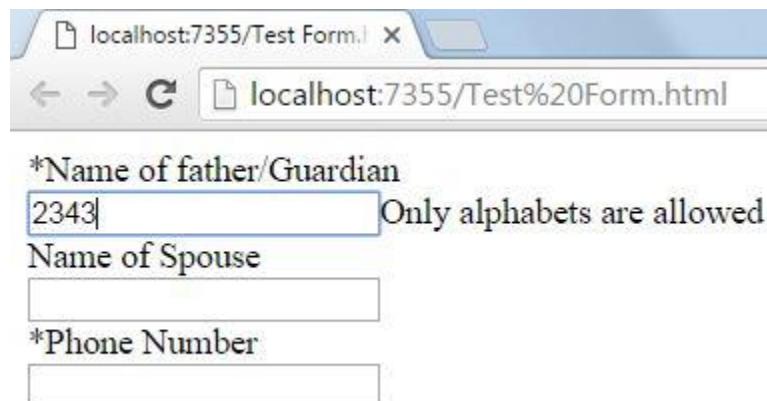


*Name of father/Guardian
 Please enter father name.

Name of Spouse

*Phone Number

Even if you enter the wrong value then also it will not allow you to leave the textbox.



*Name of father/Guardian
 Only alphabets are allowed

Name of Spouse

*Phone Number

MVC Routing With AngularJS Routing

Introduction

In this I am going to explain **how does MVC Routing and Angular Routing work when used together.**

If you have worked on MVC and Angular both but not together then you must have wondered that what will happen when both are used together. As both are having controllers, both are having routings and many more things.

It has become a hot topic for interviewers as well, as they try to mix things between Angular and MVC, just for e.g. they might ask what is the difference between AngularJS Controller and MVC Controller or between MVC routing and Angular Routing, or they might ask how you can use both the routings together and which one will work first. (OOPS! I am discussing some hot questions.)

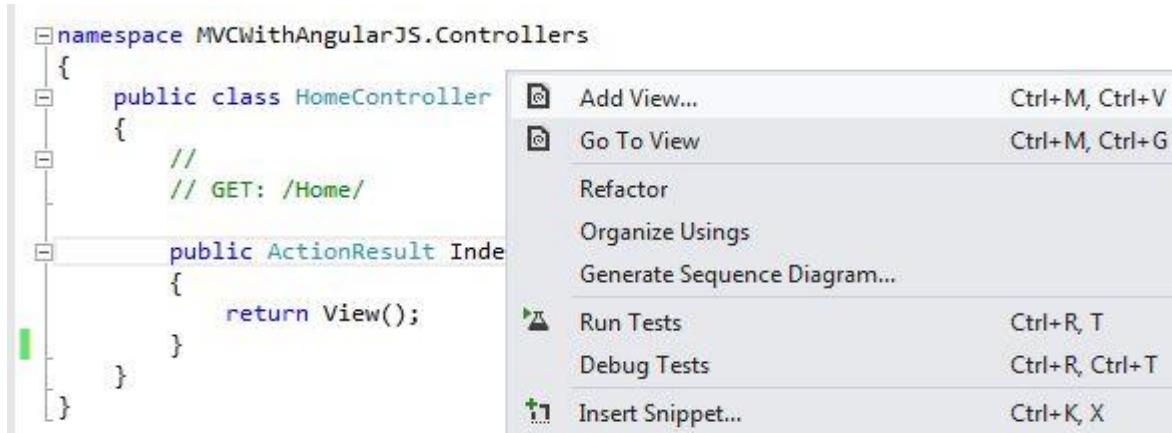
I am not going to answer all these questions as this article is not about interview questions, here I'll only answer the last question i.e. **How to use them together.** To explain this I created a sample application using MVC and AngularJS.

Step 1 - Firstly, an empty MVC application was created, in the application I add a controller file to the controller folder and just created a simple ActionResult which is calling the view Index.

```
namespace MVCWithAngularJS.Controllers
{
    public class HomeController : Controller
    {
        //
        // GET: /Home/

        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Step 2 - Right click on Index() and add click on **Add View**, this will create a folder inside Views folder with same name as of controller and in that folder a cshtml file will be created with name Index.



Step 3 - After this I worked on the cshtml page, here I called the Angular library, provided ng-app and ng-controller, and at last created a div where ng-view is applied, ng-view will make this div a container where we can call some other pages.

```

<!DOCTYPE html>
<html ng-app="angularModule">
<head>
    <script src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.20/angular.min.js"></script>
    <script src="~/Scripts/AngularController.js"></script>
</head>
<body>
    <form ng-controller="indexController">
        <h2>I am Parent Page</h2>
        <fieldset>
            <legend>Getting data from controller</legend>
            <input type="text" ng-model="name" />
        </fieldset>
        <a href="/#/firstPage">Call First Page</a>
        <br />
        <a href="/#/secondPage">Call Second Page</a>
        <div ng-view></div>
    </form>
</body>
</html>

```

But prior to the ng-view div you can see that I had created hyperlinks on which some path is defined which is having "#(hash)" with them, this # is playing the major role in this article, you can say that **# is the HERO** here, but why so! You will get the answer at the end of article.

Step 4 - We have created a container which will hold some pages but which pages! For that I again went back to Controller folder and added a new controller named as "RouteDemoController".

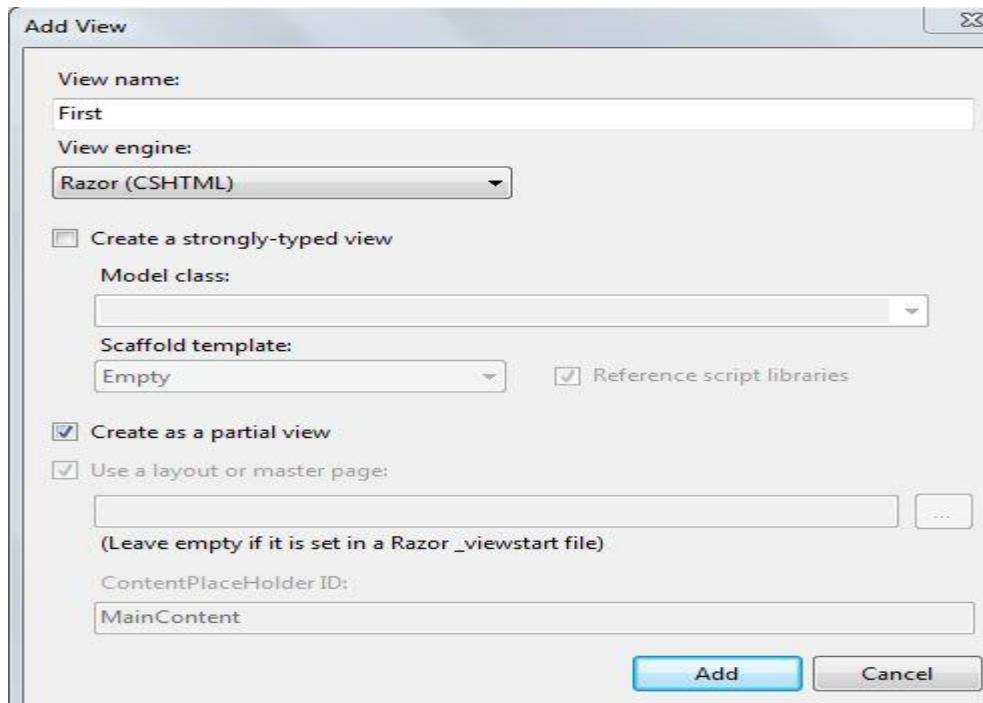
In this controller I created two ActionResult "First and Second".

```
namespace MVCWithAngularJS.Controllers
{
    public class RouteDemoController : Controller
    {
        //
        // GET: /RouteDemo/

        public ActionResult First()
        {
            return View();
        }

        public ActionResult Second()
        {
            return View();
        }
    }
}
```

Again I right clicked on these Action Results and added View Pages, but this time these pages are partial pages.



Step 5 - After this I added a JavaScript file in Scripts folder, this file will work for AngularJS part.

Here I created a controller named "indexController" in which a scope object is defined, after creating the controller I came to main part i.e. Routing part.

```
var angularModule = angular.module("angularModule", ['ngRoute']);
angularModule.controller("indexController", function ($scope) {
    $scope.name = "Anubhav";
})
angularModule.config(function ($routeProvider) {
    $routeProvider.
    when('/firstPage', {
        templateUrl: 'routedemo/first',
        controller: 'routeDemoFirstController'
    }).
    when('/secondPage', {
        templateUrl: 'routedemo/second',
        controller: 'routeDemoSecondController'
    })
})
```

You can see that in the routing section I had checked the URL and according to URL template and controller the template is assigned. These templates are those partial pages which we had created earlier, and these controllers are added after the routing is completed.

```
angularModule.controller("routeDemoFirstController", function ($scope) {
    $scope.FirstName = "Anubhav";
    $scope.LastName = "Chaudhary";
})
angularModule.controller("routeDemoSecondController", function ($scope) {
    $scope.MobileNumber = "123456";
    $scope.EmailID = "anu@test.com";
})
```

Step 6 - After completing the routing part I went to the partial pages and bind the scope values to some textboxes.

First.cshtml:

```
<div>
    <h3>First Page Displayed</h3>
</div>
<div>
    <fieldset>
        <legend>First Page Displayed</legend>
        First Name:
        <input type="text" ng-model="FirstName" />
        <br />
        Last Name:
        <input type="text" ng-model="LastName" />
    </fieldset>
</div>
```

Second.cshtml:

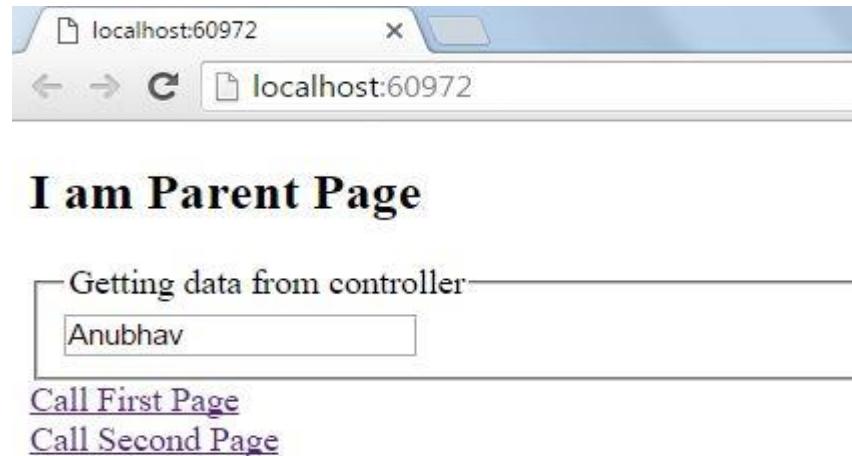
```
<div>
    <h3>Second Page Displayed</h3>
</div>
<div>
    <fieldset>
        <legend>Second Page Displayed</legend>
        Mobile Number:
        <input type="text" ng-model="MobileNumber" />
    </fieldset>
</div>
```

```

<br />
Email ID:
<input type="text" ng-model="EmailID" />
</fieldset>
</div>

```

Step 7 - Now our application is created and it's time to execute it.



I am Parent Page

Getting data from controller

[Call First Page](#)

[Call Second Page](#)

On running the application you can see our Index page is loaded, scope value is assigned in the textbox and two links are provided. When I click on any of these links, this happens:



I am Parent Page

Getting data from controller

[Call First Page](#)

[Call Second Page](#)

First Page Displayed

First Page Displayed

First Name:

Last Name:

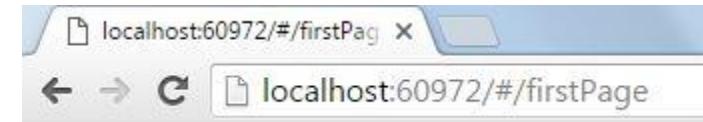
and,



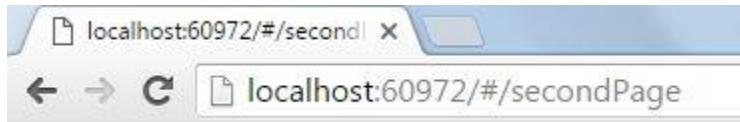
The screenshot shows a browser window with the URL `localhost:60972/#/secondPage`. The page title is "I am Parent Page". Below the title, there is a section titled "Getting data from controller" containing a text input field with the value "Anubhav". Underneath this section are two links: "Call First Page" and "Call Second Page". Below these links, there is another section titled "Second Page Displayed" containing two text input fields: "Mobile Number: 123456" and "Email ID: anu@test.com".

Partial View is loaded for the link which we had clicked and it's textboxes are having the same value which we provided in the controller of these pages (remember routing part of Angular!).

Now just have a look at the URL:



and,



Now I'll tell why # is HERO here, **prior to "##" MVC routing works and after the "##" Angular routing works** i.e. first of all MVC checks which controller and which ActionResult is called and accordingly it takes you to that View page and after reaching that page when you clicked on any link our routing checks which partial view/page is called and accordingly supplies the page and controller, so # was making things separate for MVC and Angular. I hope now interview question is solved.

Conditional Routing In AngularJS

Introduction

You might be familiar with AngularJS Routing, and might have used it as well, but what if that routing needs to be applied on fulfillment of certain conditions? For those cases this article will help you.

To explain the conditional routing I am creatinghave created a sample application.

You might have read my previous topic i.e. MVC Routing with Angular Routing. The application which was created in that article is also used here; I am just modifying it to my needs.

Step 1 - Firstly, I created a new View which will be used as a login page.

On this page I created two textboxes and a button, which will help to identify user authenticity. Apart from that I have created a div in which some message is written for non-authentic and non-authorized users, and one more div where some links and a container div is created.

```
<!DOCTYPE html>
<html ng-app="angularModule">
<head>
  <script src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.20/angular.min.js"></script>
  <script src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.20/angular-route.min.js"></script>
  <script src="~/Scripts/AngularController.js"></script>
</head>
<body>
  <form ng-controller="indexController">
    <div>
      <fieldset>
        <legend>Login Demo</legend>
        User Name:
        <input type="text" ng-model="authentic.name" />
        <br />
        Password:
        <input type="password" ng-model="authentic.pwd" />
        <br />
        <br />
    </div>
  </form>
</body>
```

```

        <input type="button" value="Submit" ng-click="CheckUser(authentic)" />
    </fieldset>
</div>
<div ng-hide="AuthenticUser()">
    <a href="/#/firstPage">Call First Page</a>
    <br />
    <a href="/#/secondPage">Call Second Page</a>
    <div ng-view></div>
</div>
<div ng-hide="CheckNotaValidUser()">
    <h2>You are not an authorized user</h2>
</div>
</form>
</body>
</html>

```

Step 2 - After this I worked on the Angular part, I created a new controller, where first of all a click function is created. In that function the models value is coming in "authentic" object.

```

var angularModule = angular.module("angularModule", ['ngRoute']);
angularModule.controller("indexController", function ($scope, $rootScope)
{
    $scope.CheckUser = function (authentic)
    {
        if (authentic.name == "Anubhav" && authentic.pwd == "Chaudhary")
        {
            $rootScope.validUser = true;
            $rootScope.userType = "Admin";
            $rootScope.NotaValidUser = false;
        }
        else if (authentic.name == "Mohit" && authentic.pwd == "Chaudhary")
        {
            $rootScope.validUser = true;
            $rootScope.userType = "Normal";
        }
        else
        {
            $rootScope.validUser = false;
            $rootScope.NotaValidUser = true;
        }
    };
};

```

According to the values entered in textboxes, some decisions are made, and accordingly some values are provided to \$rootScope objects, which will help us while creating conditional routing.

You can see that the two divs which are created after the login div shown hide on a conditional basis, and for that, I have put functions in them which will provide them boolean value, and according to those values, these divs will be shown or hide. So, let's create these two functions; these will be created in the same Controller wherethe click function was created.

```
$scope.CheckNotaValidUser = function ()
{
    if ($rootScope.NotaValidUser != undefined)
    {
        if ($rootScope.NotaValidUser == true)
        {
            return false;
        }
        else
        {
            return true;
        }
    }
    else
    {
        return true;
    }
}
$scope.AuthenticUser = function ()
{
    if ($rootScope.validUser == true)
    {
        return false;
    }
    else
    {
        return true;
    }
};
```

Step 3 - Now let's create routing, which will work on the click of hyperlinks available in third div.

```
angularModule.config(function ($routeProvider)
{
    $routeProvider.
    when('/firstPage',
    {
        templateUrl: 'routedemo/first',
        controller: 'routeDemoFirstController'
    }).
    when('/secondPage',
    {
        templateUrl: 'routedemo/second',
        controller: 'routeDemoSecondController'
    })
})
angularModule.controller("routeDemoFirstController", function ($scope)
{
    $scope.FirstName = "Anubhav";
    $scope.LastName = "Chaudhary";
})
angularModule.controller("routeDemoSecondController", function ($scope)
{
    $scope.MobileNumber = "123456";
    $scope.EmailID = "anu@test.com";
})
```

Step 4 - Now comes the main part, i.e., placing the condition for routing. For this I have used "**\$locationChangeStart**" of Angular. This function will hit as soon as user clicks on any hyperlink or goes to some other page. It will work prior to anything else, so here we can check whether the user is authentic/authorized or not to see some values or to visit some page. If he's not, then we can redirect him to some other page, or we can show him some message, or we can also just let him stay on the current page.

```
angularModule.run(function ($rootScope, $location)
{
    $rootScope.$on("$locationChangeStart", function (event, next, current)
    {
        if ($rootScope.userType != "Admin")
        {
            $rootScope.NotValidUser = true;
            $location.path("");
        }
        else {
            $rootScope.NotValidUser = false;
```

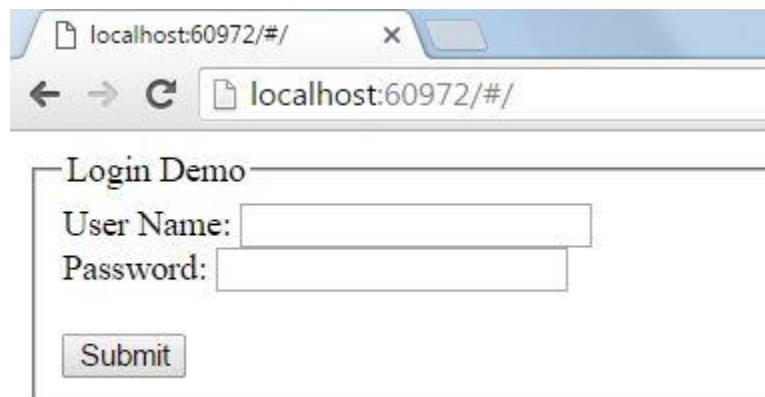
```
        }
        if ($rootScope.userType == undefined)
        {
            $rootScope.NotAValidUser = false;
        }
    });
})
```

You can see that I have checked whether the user is "Admin" or not; if not, then actions are taken accordingly. I will just show him some message and will not allow him to see any other content.

You can redirect the user to some other page using **\$location.path()**. For example, `$location.path("/Dashboard");`

Now our sample application is created and it's time to run it.

Output: On running the application two textboxes appear with a submit button.



Firstly, I provided the Admin's user name and password and checked whether everything is working fine or not, you can see that two hyperlinks appeared:

localhost:60972/#/

← → C localhost:60972/#/

Login Demo

User Name: Anubhav

Password:

Submit

[Call First Page](#)

[Call Second Page](#)

As I click on any of them partial view is shown:

localhost:60972/#/second

← → C localhost:60972/#/secondPage

Login Demo

User Name: Anubhav

Password:

Submit

[Call First Page](#)

[Call Second Page](#)

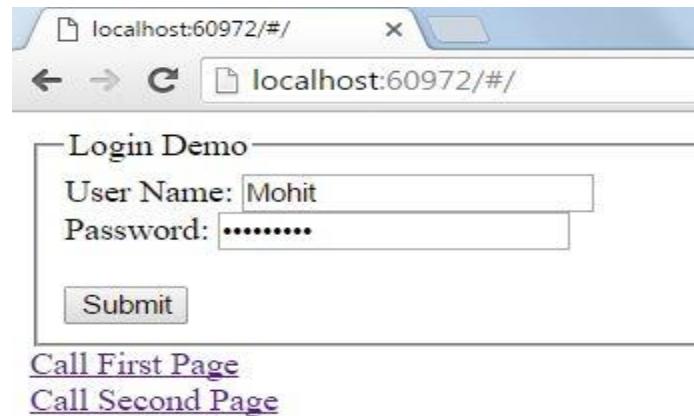
Second Page Displayed

Second Page Displayed

Mobile Number: 123456

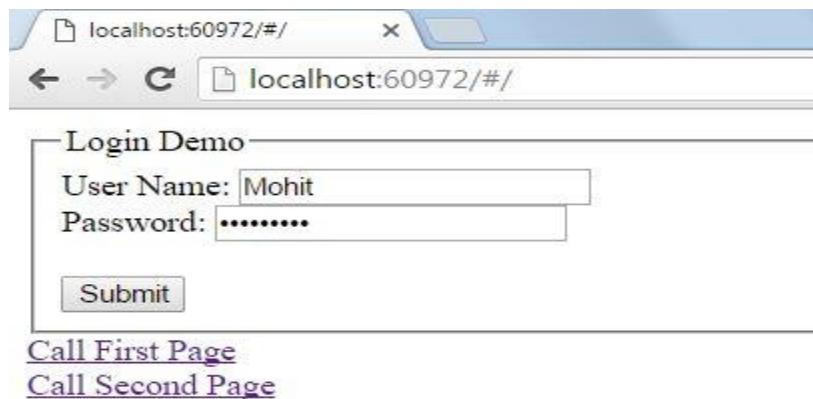
Email ID: anu@test.com

After this I provided some other user name who has limited authorization and again hyperlinks appeared:



The screenshot shows a web browser window with the URL `localhost:60972/#/`. The page title is "Login Demo". It contains a form with fields for "User Name" (containing "Mohit") and "Password" (containing "*****"). Below the form is a "Submit" button. At the bottom of the page are two links: "Call First Page" and "Call Second Page".

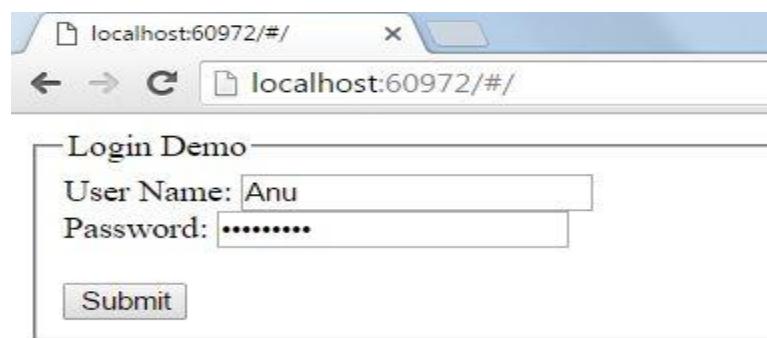
But if I click on any of the links then a message is displayed, then:



The screenshot shows a web browser window with the URL `localhost:60972/#/`. The page title is "Login Demo". It contains a form with fields for "User Name" (containing "Mohit") and "Password" (containing "*****"). Below the form is a "Submit" button. At the bottom of the page are two links: "Call First Page" and "Call Second Page". A large bold message "You are not an authorized user" is displayed below the links.

You are not an authorized user

If any non-existing user tries to enter, then nothing is shown to him:



The screenshot shows a web browser window with the URL `localhost:60972/#/`. The page title is "Login Demo". It contains a form with fields for "User Name" (containing "Anu") and "Password" (containing "*****"). Below the form is a "Submit" button. A large bold message "You are not an authorized user" is displayed below the links.

You are not an authorized user

Promise In AngularJS

Introduction

Promise is like giving some work to someone and he promises you that the work will be complete. Now until the work is done you can prepare yourself for three situations i.e. he'll do it in a perfect manner, he'll do it but it will not be useful for you, and lastly he has not done it due to some reason.

I have created a sample application to explain it.

Step 1 - Firstly, I added a HTML page which will be used to call the controller, and show the custom output.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="angular.js"></script>
    <script src="AngularScript.js"></script>
</head>
<body ng-app="angularApp">
    <form ng-controller="angularController">
        <h2><label ng-bind="Member"></label></h2>
    </form>

</body>
</html>
```

Here, I created a h2 tag which will show the result coming due to response from promise.

Step 2 - After creating the HTML, I worked on the Angular part and for this I created a separate JavaScript file.

```

var angularApp = angular.module('angularApp', []);
angularApp.controller('angularController', function ($scope, angularFactory) {
    angularFactory.checkValues().then(function (data) {
        var UserType = JSON.parse(data.d);
        if (UserType.Name == "Anubhav") {
            $scope.Member = "You are Admin";
        }
        else {
            $scope.Member = "You are not Admin";
        }
    }, function (error) {
    });
});

angularApp.factory('angularFactory', function ($http, $q) {
    return {
        checkValues: function () {
            dataArr = [];
            dataArr.push(new Object());
            dataArr[0]["Name"] = "Anubhav";
            return $http({
                url: 'WebService1.asmx/Save_Session',
                dataType: 'json',
                method: 'POST',
                data: "{array:" + JSON.stringify(dataArr) + "}",
                headers: {
                    "Content-Type": "application/json"
                }
            }).then(function (resp) {
                if (typeof resp.data === 'object') {
                    return resp.data;
                } else {
                    return $q.reject(response.data);
                }
            }, function (err) {
                return $q.reject(err.data);
            });
        });
    };
});

```

Here, firstly I created a module and name it "**angularApp**".

After this I created a controller "**angularController**" in which factory method is called. Here you can see that I have used ".**then**;" it works only when our factory method returns something. Inside that first function is the success function and the second one is error function.

After Controller, I created factory where I made a HTTP POST call to WebService Method, I passed some dummy data in an object. Here also you can see that I have used ".**then**" which clearly means that functions inside "**then**" will only execute when something is returned from WebService, if object is returned then it is passed to controller, otherwise deferred state is rejected.

Step 3 - At the end I created a WebMethod in WebService.

```
[WebMethod]
public object Save_Session(List<Object> aray)
{
    string text = "";
    try
    {
        foreach (Dictionary<string, object> item in aray)
        {
            text = new JavaScriptSerializer().Serialize(item);
        }
        return text;
    }
    catch (Exception ex)
    {
        return text;
    }
}
```

You can see that from here I am also sending the object, which was needed in the success function. You can modify all the codes according to your needs, it's just a simple method.

Now let's run our application and see the output.

Output: On running the application, just open the console and check that controller is hit.

Sources Network Timeline Console Profiles Resources Audits

AngularHTML.html angular.js AngularScript.js x

① Serving from the file system? Add your files into the workspace. [more](#)

```

1 var angularApp = angular.module('angularApp', []);
2 angularApp.controller('angularController', function ($scope, angularFactory) {
3     debugger;
4     angularFactory.checkValues().then(function (data) {
5         var UserType = JSON.parse(data.d);
6         if (UserType.Name == "Anubhav") {
7             $scope.Member = "You are Admin";
8         }
9         else {
10            $scope.Member = "You are not Admin";
11        }
12    }, function (error) {
13        debugger;
14    });
15 });
16 });
17 angularApp.factory('angularFactory', function ($http, $q) {
18     return {
19         ...
20     };
21 });
22 
```

{ } Line 3, Column 5

From controller, the **factory** method is called and from there sample object is send to the WebService:

Sources Network

AngularHTML.html

① Serving from the file system? Add your files into the workspace. [more](#)

```

1 var angularApp = angular.module('angularApp', []);
2 angularApp.controller('angularController', function ($scope, angularFactory) {
3     debugger;
4     angularFactory.checkValues().then(function (data) {
5         var UserType = JSON.parse(data.d);
6         if (UserType.Name == "Anubhav") {
7             $scope.Member = "You are Admin";
8         }
9         else {
10            $scope.Member = "You are not Admin";
11        }
12    }, function (error) {
13        debugger;
14    });
15 });
16 });
17 angularApp.factory('angularFactory', function ($http, $q) {
18     return {
19         ...
20     };
21 });
22 
```

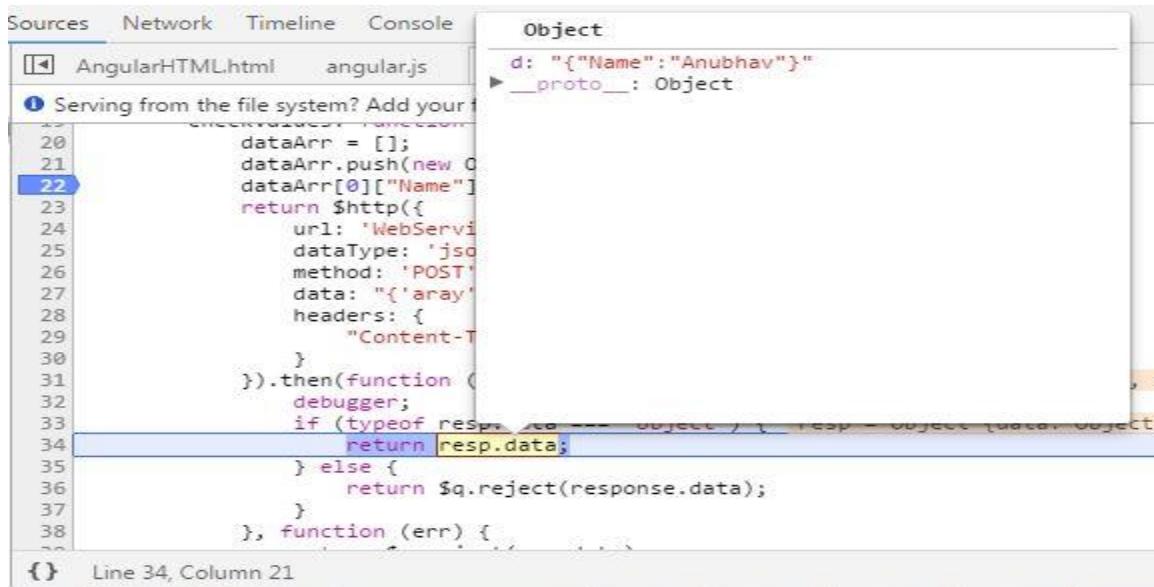
dataArr[0]["Name"] = "Anubhav";

return \$http({

url: 'WebService1.asmx/Save_Session',
 dataType: 'json',
 method: 'POST',

{ } Line 23, Column 13

Now from Web Service again object is send which is retrieved in first function inside "then"



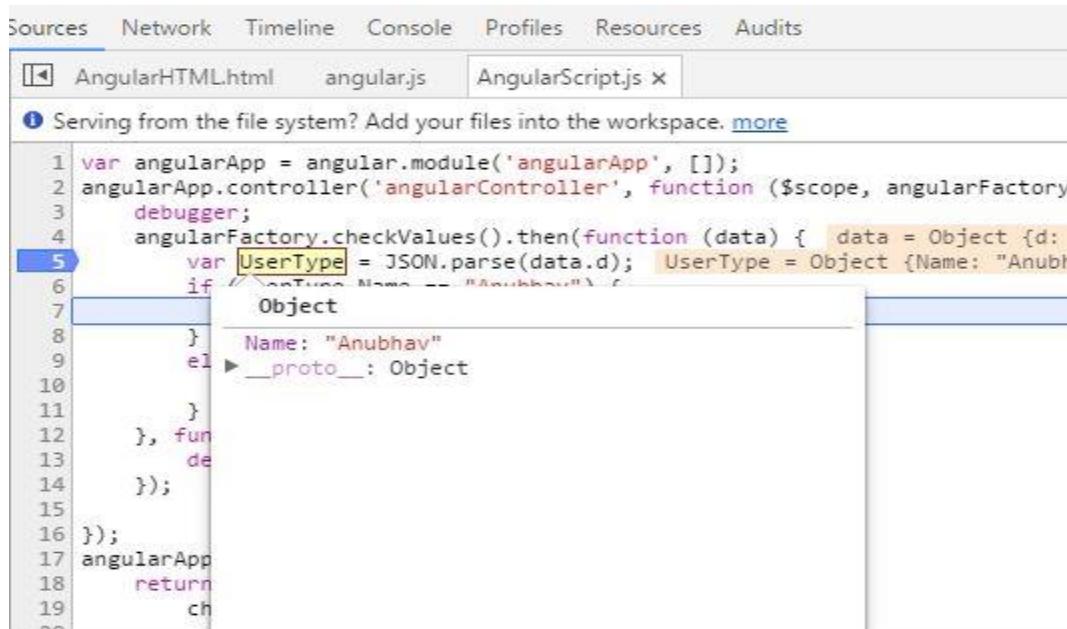
```

Sources Network Timeline Console
AngularHTML.html angular.js
① Serving from the file system? Add your files into the workspace. more
20     dataArr = [];
21     dataArr.push(new Object());
22     dataArr[0]["Name"] = "Anubhav";
23     return $http({
24         url: 'WebService/GetUser',
25         dataType: 'json',
26         method: 'POST',
27         data: {"array": dataArr},
28         headers: {
29             "Content-Type": "application/json"
30         }
31     }).then(function (resp) {
32         debugger;
33         if (typeof resp === 'object') {
34             return resp.data;
35         } else {
36             return $q.reject(response.data);
37         }
38     }, function (err) {
39     });

```

{ } Line 34, Column 21

Now this data is returned to controller and there some conditions are checked for this returned data.

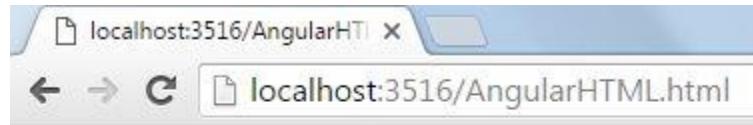


```

Sources Network Timeline Console Profiles Resources Audits
AngularHTML.html angular.js AngularScript.js
① Serving from the file system? Add your files into the workspace. more
1 var angularApp = angular.module('angularApp', []);
2 angularApp.controller('angularController', function ($scope, angularFactory) {
3     debugger;
4     angularFactory.checkValues().then(function (data) {
5         var UserType = JSON.parse(data.d);
6         UserType = Object {Name: "Anubhav"};
7         if (UserType === Object) {
8             $scope.UserType = UserType;
9         } else {
10            $scope.UserType = UserType.Name;
11        }
12    }, function (err) {
13        $scope.error = err.message;
14    });
15 });
16 angularApp
17     .factory('angularFactory', [
18         '$q',
19         '$http',
20         'checkValues',
21         angularFactory
22     ]);

```

According to the conditions fulfilled data is shown on UI.



You are Admin

Different Kinds Of Scopes In Custom Directives

Introduction

While we create a custom directive, there is a possibility that you will want to use the scopes available in the parent controller. You can use the scopes of the parent controller in different levels. This article will help you choose in between those levels.

I created a sample application which will help you understand it in an easy manner.

Step 1 - Firstly, I added a HTML page where our custom control is created.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="angular.js"></script>
    <script src="AngularScript.js"></script>
</head>
<body ng-app="angularApp">
    <form ng-controller="angularController">
        <fieldset>
            <legend>I am binding from controller
            </legend>
            <input type="text" ng-model="EnterName" />
        </fieldset>
        <br />
        <fieldset>
            <legend>I am binding from directive
            </legend>
            <custom-directive></custom-directive>
        </fieldset>
    </form>

</body>
</html>
```

In this code you can see I had bound the form tag with a controller named "**angularController**," this controller will be created in upcoming steps.

In this form I used two fieldsets to distinguish between normal controller binding and custom directive. In the firstfieldset I created an input control which is bound to a model named "**EnterName**."

In the secondfieldset I created a custom control named "**custom-directive**," using this control I'll create the custom directive.

Step 2 - Now it's time to work on the angular part. I added a JavaScript file to my application in which I had written my angular code.

Firstly, I created a module and a controller.

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

angularApp.controller('angularController', function ($scope) {
    $scope.EnterName = "Anubhav";
});
```

Now it's time to create our custom directive, but before creating it, let's first know how scopes will be used in this directive or what level will be used here.

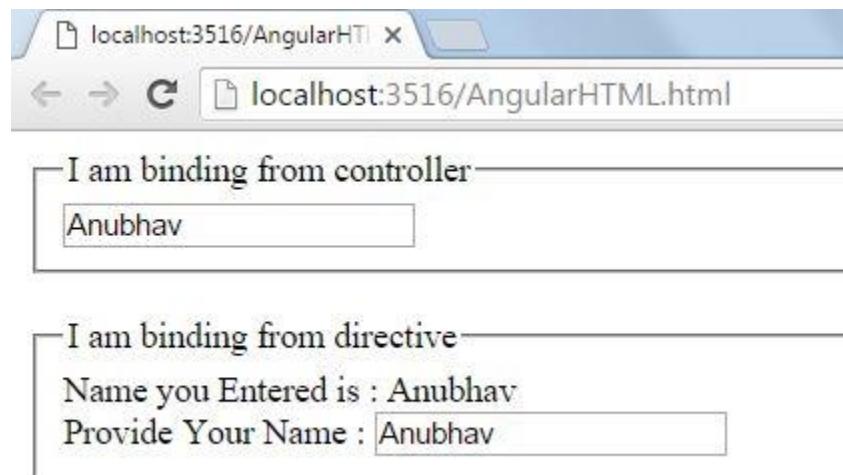
Scope: false

So, in this directive you will be able to access the scopes completely without having any kind of restriction. Let's first see the code and after that I'll explain it.

```
angularApp.directive("customDirective", function () {
    return {
        restrict: "E",
        scope: false,
        template: "<div>Name you Entered is : {{EnterName}}</div>" +
        "Provide Your Name : <input type='text' ng-model='EnterName' />"
    };
});
```

You can see that I wrote "scope : false," it means **custom directive will not have its own new scope and it'll inherit its parent scope completely**. So whatever model you have defined in your controller can be easily used here, just as I have used in the template. One more important thing with "scope : false" is that if you make changes in scope data then that will be reflected in both controller and in directive as well.

Let's see the output to clearly understand it.



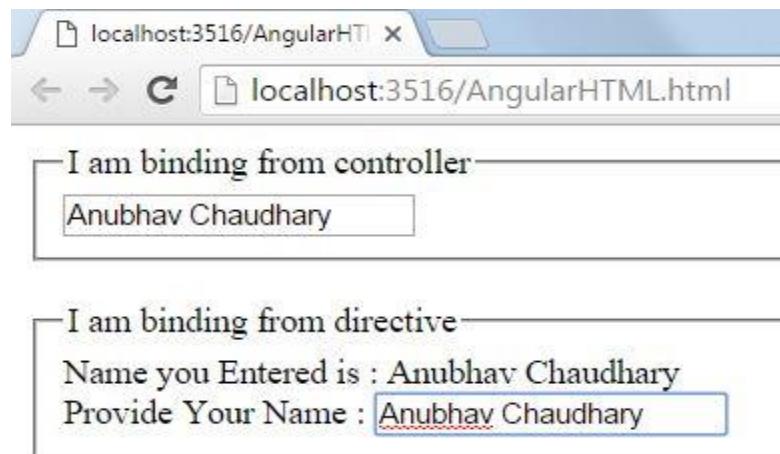
The screenshot shows a web browser window with the URL `localhost:3516/AngularHTML.html`. It displays two identical sections of HTML code. Both sections contain a label "I am binding from controller" followed by a text input box containing the value "Anubhav".

```
I am binding from controller  

```

```
I am binding from directive  
Name you Entered is : Anubhav  
Provide Your Name : 
```

You can see that both the textboxes and label have the predefined data, now if I change the data in any of the textbox then changes will occur in all the controls which are bound to this model.



The screenshot shows a web browser window with the URL `localhost:3516/AngularHTML.html`. It displays two identical sections of HTML code. The first section is labeled "I am binding from controller" and contains a text input box with the value "Anubhav Chaudhary". The second section is labeled "I am binding from directive" and contains a label "Name you Entered is : Anubhav Chaudhary" and a text input box with the value "Anubhav Chaudhary".

```
I am binding from controller  

```

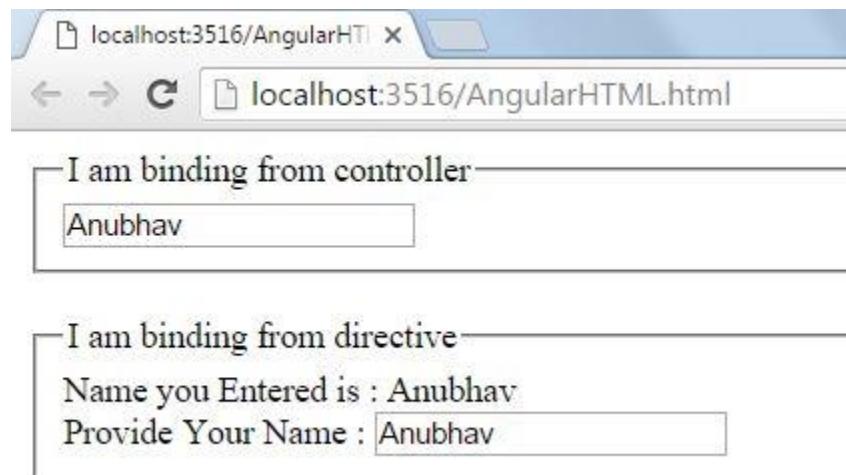
```
I am binding from directive  
Name you Entered is : Anubhav Chaudhary  
Provide Your Name : 
```

Scope: true

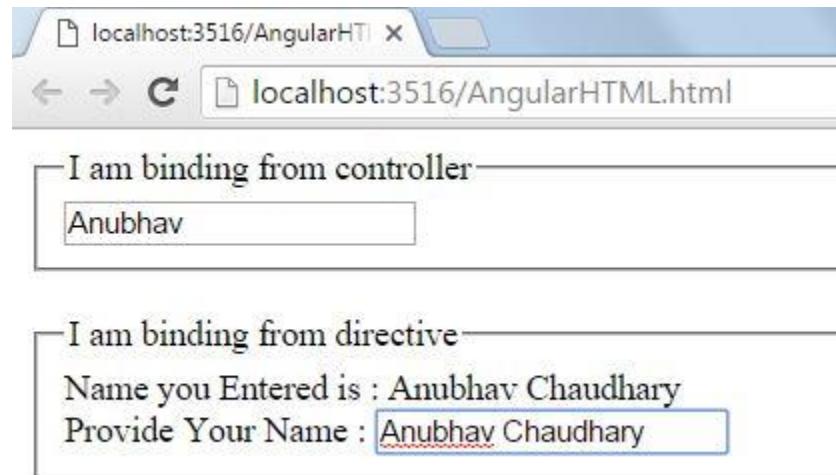
It might be possible that you want to **use the value of parent scopes for one time only** i.e. at initial stage but after that you don't want your directive scope to get updated if there is a change at parent scope or vice versa, then in that case just provide "**scope : true.**"

```
angularApp.directive("customDirective", function () {
  return {
    restrict: "E",
    scope: true,
    template: "<div>Name you Entered is : {{EnterName}}</div>" +
      "Provide Your Name : <input type='text' ng-model='EnterName' />"
  };
});
```

Now let's again run the application and check what is happening.



As the page loads all the controls get their initial value which was predefined in the controller, but if I change the data in second textbox i.e. at custom directive level, then only those controls will get updated which are created using custom directive and everyone else will remain the same.



Scope: {}

Scope: true was providing you a new scope but after the initial level, what if you do not want any kind of interference between controller and directive i.e. you do not want to use any value from controller? Therefore, in that case make **isolated scope** i.e. "scope : {}"

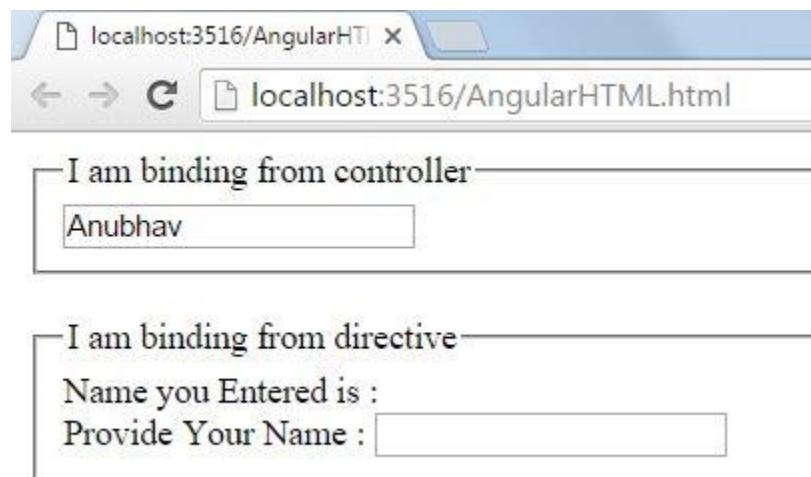
```
var angularApp = angular.module('angularApp', []);

angularApp.controller('angularController', function ($scope) {
    $scope.EnterName = "Anubhav";
});

angularApp.directive("customDirective", function () {
    return {
        restrict: "E",
        scope: {},
        template: "<div>Name you Entered is : {{EnterName}}</div>" +
        "Provide Your Name : <input type='text' ng-model='EnterName' />"
    };
});
```

You can see that still in the template I bound the controls with the same model to which they were earlier bound but now this model will not have any impact from previous one because they are defined in a new scenario.

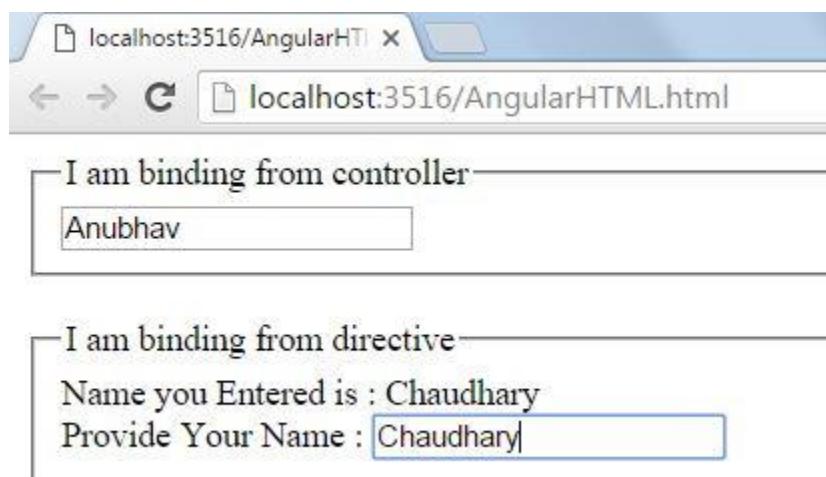
Let's again run the application and check what's happening.



I am binding from controller
Anubhav

I am binding from directive
Name you Entered is :
Provide Your Name :

You can see that only first textbox is getting the value but our label and second textbox is empty. But if I try to write any value in the second textbox then label also get updated because it's bound to the same model to which second textbox is bound.



I am binding from controller
Anubhav

I am binding from directive
Name you Entered is : Chaudhary
Provide Your Name : Chaudhary

Isolated Scopes In Custom Directives

Introduction

In my previous topic I explained about "Different Kind of Scopes in Custom Directives", as i promised you previously that I'll explain **Isolated Scopes in Custom Directives**, so here it is.

I am using previous sample application to explain Isolated Scopes.

Step 1 - As we had already created a sample application so we can directly go to the controller of Angular. In the controller I am adding a new property into the scope and a new function as well which will be called on button click.

```
var angularApp = angular.module('angularApp', []);
angularApp.controller('angularController', function ($scope, angularFactory) {
    $scope.EnterName = "Anubhav";
    $scope.ProvideName = "Anubhav Chaudhary";
    $scope.MakeCall = function () {
        alert("Button Clicked");
    }
});
```

Step 2 - After this it's time to work on the directive where real changes need to be done.

```
angularApp.directive("customDirective", function () {
    return {
        restrict: "E",
        scope: {
            text: "@",
            doublebind: "=",
            call: "&"
        },
        template: "<div>Name you Entered is : {{text}}</div> +
        "Provide Your Name : <input type='text' ng-
model='text' /> <br /> I am for two way binding : <input type='text' ng-
model='doublebind' /> <br /> <input type='button' ng-click='call()' value='Click Me' />" +
    };
});
```

You can see that I had made some changes in scope of this directive. I had provided some symbols and these symbols are known as "**Prefix**".

Prefix-

Prefix are important thing in isolated scopes, **prefix assign the values of properties and methods from controller scope to directive scope**. If nothing is provided after the prefix then it will search for the attribute which is having the same property name on directive's html element, but if something is provided after the prefix then it will search for that attribute in the directive's html element.

Let's see what does these prefix symbols mean:

@: It is used when we want to provide one way binding i.e. if we change in the controller scope then same changes will be applied in the directive scope but vice versa is not allowed.

= : It's used when we want to provide two way binding i.e. if we change either in controller scope or directive scope then changes will be applied on both.

&: It is used when we want to assign any method from controller scope to directive scope.

Here you can see that I have not provided anything after the prefix, so it'll search for the same property name.

In the template you can see that I have bind the different textbox and button with these properties, so some of them will show one way binding, some will show two way and button will call the method which is defined in controller.

Step 3 - It's time to work on the HTML page.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="angular.js"></script>
    <script src="AngularScript.js"></script>
</head>
<body ng-app="angularApp">
    <form ng-controller="angularController">
        <fieldset>
```

```

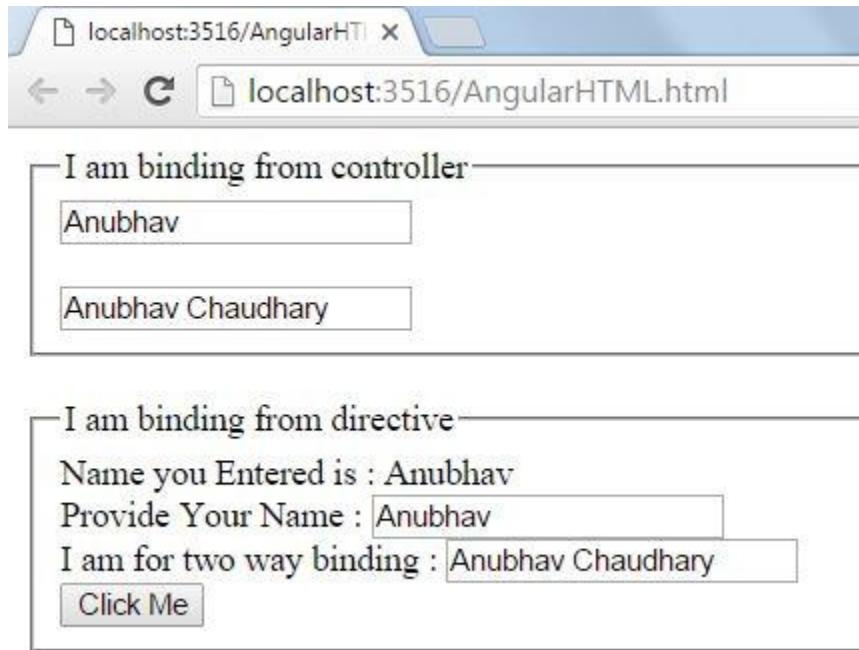
<legend>I am binding from controller
</legend>
<input type="text" ng-model="EnterName" />
<br />
<br />
<input type="text" ng-model="ProvideName" />
</fieldset>
<br />
<fieldset>
  <legend>I am binding from directive
  </legend>
  <custom-
directive text="{{EnterName}}" doublebind="ProvideName" call="MakeCall()"></custom-directive>
  </fieldset>
</form>

</body>
</html>

```

Here you can see that in the custom directive's element I had bind the attributes with the controller's scope, those which needs to be bind for one way binding are bind using {{}}.

Now let's run the application and see the output.



I am binding from controller

Anubhav Chaudhary

I am binding from directive

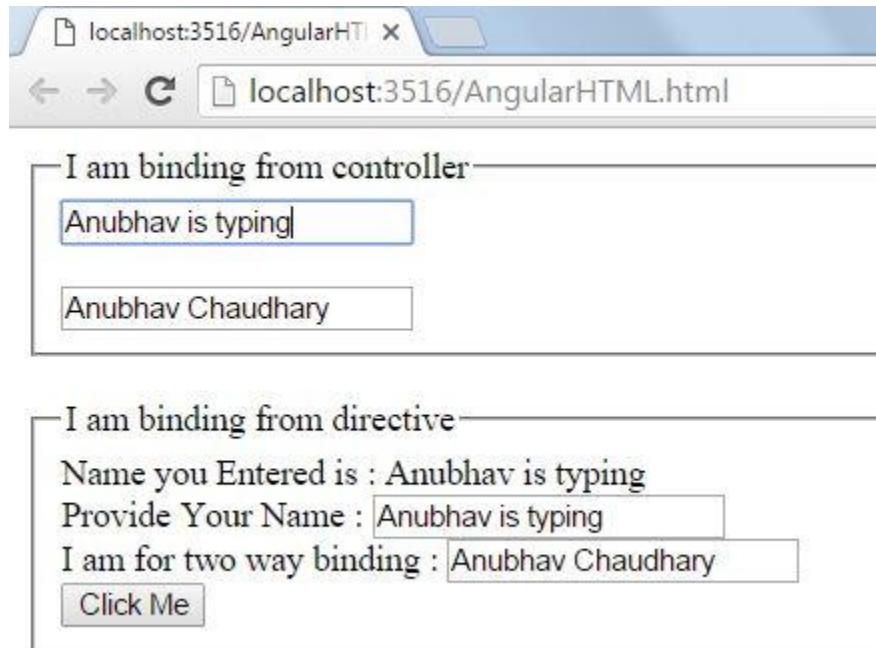
Name you Entered is : Anubhav

Provide Your Name :

I am for two way binding :

On running the application, you can see that first textboxes inside different fieldset are showing the same values because they are bind with same scope property and second textbox inside both fieldset are showing same value because these two are bind with same scope's property.

Now I am changing the value in first textbox.



You can see that changes can be seen in second fieldset controls.

But if I try to change the value in first textbox of second fieldset then corresponding label show the changes not the first textbox of first fieldset.

I am binding from controller

Anubhav is typing

Anubhav Chaudhary

I am binding from directive

Name you Entered is : Anubhav is typing but changes are not applying

Provide Your Name : it changes are not applying

I am for two way binding : Anubhav Chaudhary

Click Me

This is because here only one way binding is provided i.e. from controller to directive.

Now I am changing the value in second textbox.

I am binding from controller

Anubhav is typing

Anubhav Chaudhary is pre

I am binding from directive

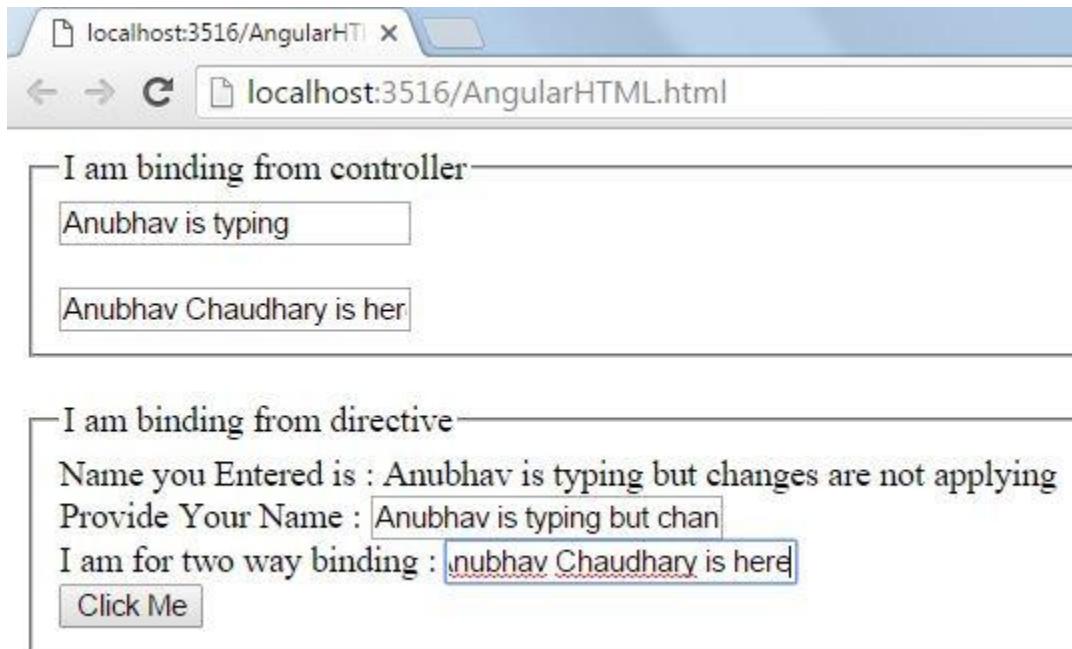
Name you Entered is : Anubhav is typing but changes are not applying

Provide Your Name : Anubhav is typing but chan

I am for two way binding : Anubhav Chaudhary is pre

Click Me

You can see that second textbox of second fieldset is showing the changed value, but what if I try to change the value from second fieldset?



I am binding from controller

Anubhav is typing

Anubhav Chaudhary is here

I am binding from directive

Name you Entered is : Anubhav is typing but changes are not applying

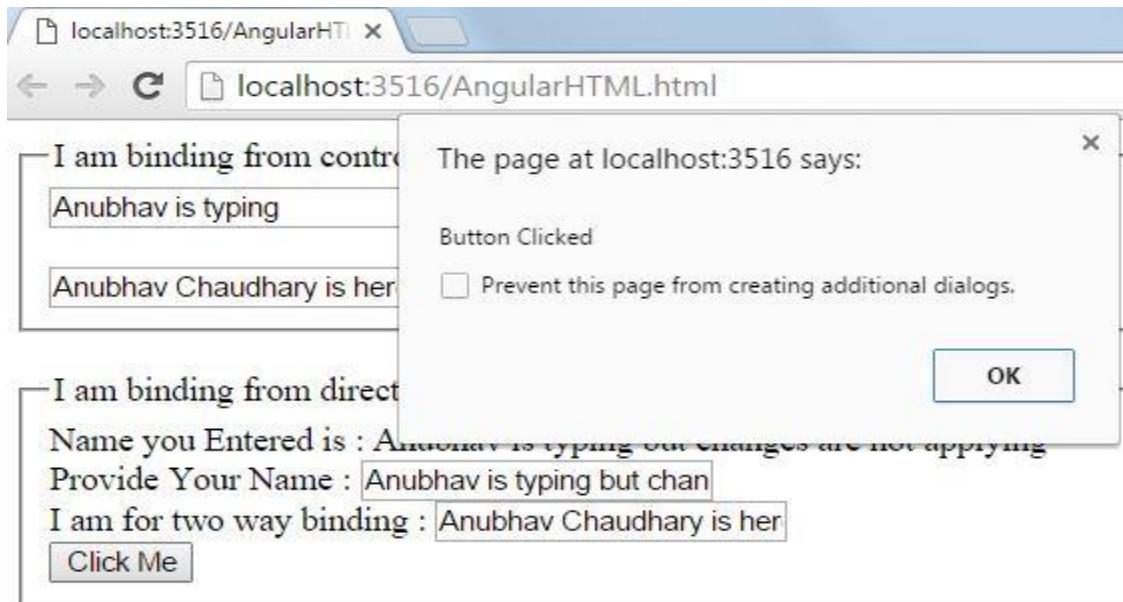
Provide Your Name : Anubhav is typing but chan

I am for two way binding : Anubhav Chaudhary is here

Click Me

You can see that changed value is displayed on the second textbox of first fieldset, this is because here two-way binding was provided.

Now I am clicking on the button which was created using the custom directive.



You can see that it has called the method inside controller due to which alert box is displayed.

Till now you have seen that how to provide isolated scopes with no text after the Prefixes. But now I am showing you how to bind when some property name is provided after the prefix.

Step 1 - Again I am working on the controller first, here I am adding a new property named "**ParentName**" and assigned some value to it.

```
angularApp.controller('angularController', function ($scope, angularFactory) {
    $scope.EnterName = "Anubhav";
    $scope.ProvideName = "Anubhav Chaudhary";
    $scope.ParentName = "Santosh Chaudhary";
    $scope.MakeCall = function () {
        alert("Button Clicked");
    }
});
```

Step 2 - After this I am making changes in the directive.

```
angularApp.directive("customDirective", function () {
    return {
        restrict: "E",
        scope: {
            text: "@",
            doublebind: "=",
```

```

parentname: "=parentName",
call: "&"
},
template: "<div>Name you Entered is : {{text}}</div>" +
"Provide Your Name : <input type='text' ng-
model='text' /> <br /> I am for two way binding : <input type='text' ng-
model='doublebind' /> <br /> Mother Name : <input type='text' ng-
model='parentname' /> <br /> <input type='button' ng-click='call()' value='Click Me' />" +
};

});

```

Here you can see that I had created a new property "**parentname**" and to this property two-way binding is assigned using the "=" prefix, but after the prefix I had provided "**parentName**".

In the template also some changes are done, a new textbox is added which is bind to this new property "parentname".

Step 3 - Finally change the HTML part.

```

<body ng-app="angularApp">
<form ng-controller="angularController">
<fieldset>
<legend>I am binding from controller
</legend>
<input type="text" ng-model="EnterName" />
<br />
<br />
<input type="text" ng-model="ProvideName" />
<br />
<br />
<input type="text" ng-model="ParentName" />
</fieldset>
<br />
<fieldset>
<legend>I am binding from directive
</legend>
<custom-directive text="{{EnterName}}" doublebind="ProvideName" parent-
name="ParentName" call="MakeCall()"></custom-directive>
</fieldset>
</form>
</body>

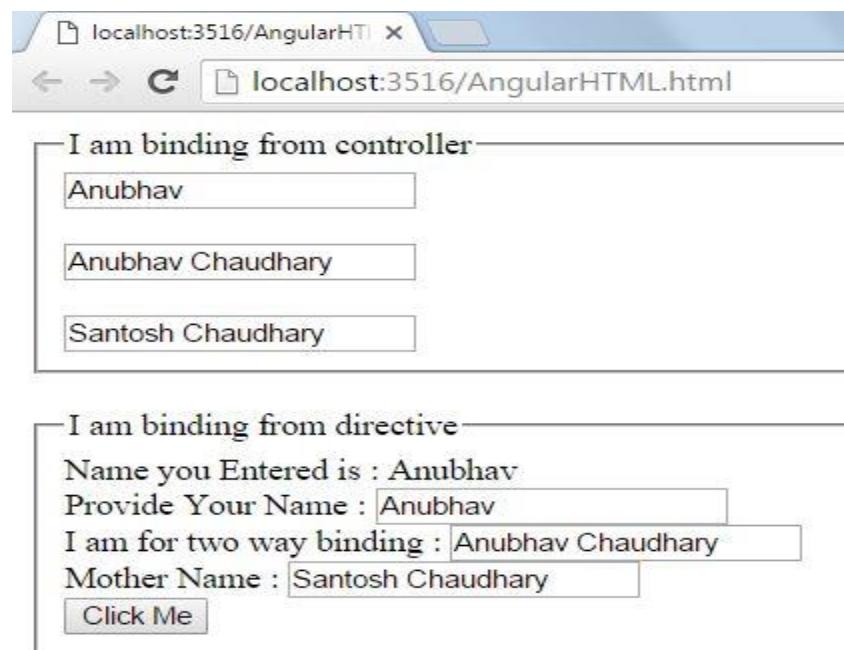
```

Here you can see that a custom attribute is added as "**parent-name**", and to this attribute Controller's scope property is assigned.

One important thing to notice is that in the HTML part "**parent-name**" is written but in the angular part "**parentName**" was provided, if you had ever created a directive then you must have already know that why it was done and for others when we are creating any custom directive, then "-" is replaced by capital character.

Now our application is created and it's ready to get executed.

Output: On running the application you will see that all the textboxes are having the values according to properties assigned to them.



Now I am making the changes in textbox which is showing the parent name in second fieldset.

localhost:3516/AngularHTI x localhost:3516/AngularHTML.html

I am binding from controller

Anubhav

Anubhav Chaudhary

Santosh Chaudhary is my

I am binding from directive

Name you Entered is : Anubhav

Provide Your Name : Anubhav

I am for two way binding : Anubhav Chaudhary

Mother Name : Santosh Chaudhary is my

You can see that in the first fieldset changes are shown that's because two way binding was provided.

Similarly you can provide the one way binding and method binding as well.

Use Multiple Modules On Same Page In AngularJS

Introduction

In this I'll explain '**How to Use Multiple Modules on the Same Page.**'

While you are working with AngularJS, you might find a situation where you are having some scopes which are available in some controller, but those controllers belong to different modules; in such a case you might have introduced new scope in controller so that you can use it. This is because **you can't declare multiple modules in "ng-app"**, this is a limitation of ng-app, it restricts you to only one module per page. To get rid of this problem you can **choose "ng-module" which is available in "angular.ng-modules"**. angular.ng-modules is a new directive which needs to be provided where you had provided the AngularJS directive, you can download it from my source code.

I am creating a sample application to show how you can use this feature.

Step 1 - Firstly, download the "**angular.ng-modules**" using the above links.

After this provide where you had provided the "angularjs" directive.

```
<head>
<title></title>
<script src="angular.js"></script>
<script src="angular.ng-modules.js"></script>
<script src="AngularController.js"></script>
</head>
```

Step 2: Now I am creating a new JavaScript file where we will declare our modules, controller, and scopes.

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

firstModule.controller('firstController', function ($scope) {
    $scope.UserName = 'Anubhav Chaudhary';
});

firstModule.controller('secondController', function ($scope) {
    $scope.MobileNumber = '00000000';
});
```

```
var secondModule = angular.module('secondModule', []);
secondModule.controller('thirdController', function ($scope) {
    $scope.EmailId = 'anu@test.com';
});
```

Here you can see that I declared two modules named "**firstModule**" and "**secondModule**," inside the firstModule I created two controllers and in those controllers I created some scope properties, inside secondModule only one controller is created and in the controller one scope property is created. To all these scopes some default value is also provided.

Step 3 - Now it's time to work on the main section i.e. HTML section.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="angular.js"></script>
    <script src="angular.ng-modules.js"></script>
    <script src="AngularController.js"></script>
</head>
<body>
    <form ng-module="firstModule,secondModule">
        <fieldset>
            <legend>We are having both the modules</legend>
            <div ng-controller="firstController">
                Provide Your Name :
                <input type="text" ng-model="UserName" />
            </div>
            <div ng-controller="secondController">
                Provide Your Mobile Number :
                <input type="text" ng-model="MobileNumber" />
            </div>
            <div ng-controller="thirdController">
                Provide Your Email ID :
                <input type="text" ng-model="EmailId" />
            </div>
        </fieldset>
    </form>
    <form ng-module="secondModule">
        <fieldset>
            <legend>I am having only second module</legend>
            <div ng-controller="thirdController">
```

```

Provide Your Email ID :  

<input type="text" ng-model="EmailId" />  

</div>  

</fieldset>  

</form>  

</body>  

</html>

```

Here you can see that I created two forms, let's firstly talk about the first form.

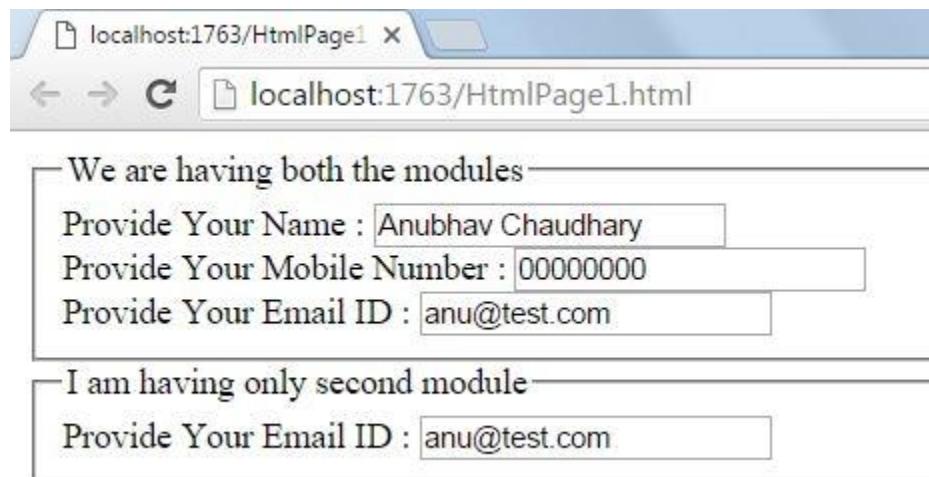
In the first form you can see that I had provided both the modules using "**ng-modules**" directive. When you are using "ng-modules" then it allows you to use multiple modules at same time, but if you don't want to use multiple modules then you can use "**ng-module**" directive.

Inside the first form all three controllers are used in three different div's and their scopes are bind to some input fields.

In the second form you can see that I have used "ng-module" because I need to use only one module here. Inside the form div is bound to a corresponding controller and an input element is bound to scope property.

Now our application is created and it's time to see the output.

Output: On running the application you will see an output like this,



All the input fields have their scope value and this means that our code has run successfully and both the modules got bound at same time.

\$templateCache In AngularJS

Introduction

In this I'll explain **\$templateCache in AngularJS**.

Templates created on your UI can be cached using \$templateCache provided by AngularJS, you can get and set the templates using \$templateCache.

Here I will create a simple sample application which will load a string in div using templateCache, but it can be used for much larger changes and I will try to create such kind of changes in my upcoming articles. So this article can be considered as "To get started with \$templateCache in AngularJS".

Step 1 - Firstly, I am adding reference for AngularJS in the head section, and binding the body section with controller which will be created in upcoming steps.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title></title>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.4.0/angular.js"></script>
</head>
<body ng-app="NewModule" ng-controller="myController">
</body>
</html>
```

Step 2 - After this I am adding a new JavaScript file, where our AngularJS code will exist.

```
var NewModule = angular.module('NewModule', []);
NewModule.controller('myController', function ($scope) {
  $scope.name = "Anubhav";
  $scope.mail = "anu@test.com";
});
```

Here you can see that I had provided some properties inside scope. But still we had not provided anything for \$templateCache, so let's create that also.

```

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

NewModule.controller('myController', function ($scope) {
    $scope.name = "Anubhav";
    $scope.mail = "anu@test.com";
});

NewModule.run(function ($templateCache) {
    $templateCache.put('loadMe.html', '<b>Hey! I am loaded</b>');
});

```

\$templateCache need to be provided inside the "run" function, here I had passed a string which will be called using the name provided before the string i.e. "loadMe.html".

Step 3 - Now we need to make changes on our HTML part so that scope values and \$templateCache value can be used.

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.4.0/angular.js"></script>
    <script src="RoutingController.js"></script>
</head>
<body ng-app="NewModule" ng-controller="myController">
    <fieldset>
        <legend>Data From Controller</legend>
        <div>
            Name: <input type="text" ng-model="name" />
            <br />
            Mail ID: <input type="text" ng-model="mail" />
        </div>
    </fieldset>
    <fieldset>
        <legend>I am getting value from template cache</legend>
        <div ng-include=""loadMe.html"">
            </div>
    </fieldset>
</body>
</html>

```

You can see that I had created a div in which cache value is included using ng-include.

Now our application is created and it's time to execute it and see the output.

Output: On running the application you can see that our string is available on the UI.

