MadewithAngular.com Will have a list of Angular based Websites
plnkr.co web development tool

What is AngularJS?

- AngularJS is a "JavaScript Library" or "JavaScript Framework", which is used to implement "MVC pattern" (Model-View-Controller Design Pattern) in the web pages.
- "JavaScript Library" means "collection of pre-defined functions".
- "JavaScript Framework" means "collection of many technologies like HTML + CSS + JavaScript + JSON + AJAX".
- "MVC" stands for "Model – View – Controller". It is a design pattern, which provides a set of rules and regulations for writing the programs.
- AngularJS is developed by Google.
- AngularJS is very popular & mostly demanded.
- AngularJS is light weight and cross-browser compatible.
- AngularJS extends HTML with new attributes.
- AngularJS is perfect for Single Page Applications (SPAs).
- AngularJS is easy to learn.

AngularJS History

- AngularJS version 1.0 was released in 2012.
- Misko Hevery, a Google employee, started to work with AngularJS in 2009.
- The idea turned out very well, and the project is now officially supported by Google.

AngularJS Introduction

- AngularJS is a JavaScript framework. It can be added to an HTML page with a <script> tag.
- AngularJS extends HTML attributes with Directives, and binds data to HTML with Expressions.
- AngularJS is a JavaScript Framework
- AngularJS is a JavaScript framework. It is a library written in JavaScript.
- AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:
- <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
- AngularJS Extends HTML
- AngularJS extends HTML with ng-directives.
- The ng-app directive defines an AngularJS application.
- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
- The ng-bind directive binds application data to the HTML view.

Problems before AngularJS

•No Separation between "HTML" and "JavaScript": Generally, we can use "normal JavaScript" or "jQuery" to perform DOM manipulations directly. In this case, we will use ID's (or css class names) in the JavaScript / jQuery code. Thus we depend on the html elements directly in the JavaScript / jQuery code. So both components can't individually developed. So this makes a maintenance problem. To overcome this problem, AngularJS is developed. In AngularJS, we solve this problem with a concept of "data bindings". The "data bindings" are very useful for "single page applications".

• No Modules: In "html, css, javascript", there is no way to divide the large project into modules. But in AngularJS, we can use "angularjs modules" concept to divide the large project into modules. "Modules" are very important for "single page applications".

• No URL Routing: In "html, css, javascript", there is no direct and easy way to create URL routing. URL routing makes the URL (address at browser's address bar) understandable and reflect the dynamic changes made in the web page. In AngularJS, we can use "AngularJS Routing" concept to implement "URL Routing". URL Routing is very important for "single page applications"

• No Unit Testing: By default, JavaScript functions are not unit-testable, as they perform DOM manipulations directly. But many AngularJS components, such as controllers, directives, services, factories etc., are unit-testable. That means we can make dummy calls to these components, and make sure those return a correct value.

Features of AngularJS

• AngularJS extends HTML language by adding "declarative code" to html. AngularJS is lets you to describe dynamic views in declarative way in a readable, expressive manner.

• AngularJS lets you implement "two-way data bindings", which makes relationship between "html tag" and "data". As this is two way data bindings, the following things happens at run time:
When the user modify the value in "html tag", the "data" will automatically changes.

• When the developer modify the value in the code, the "html tag" will automatically changes.

• AngularJS supports modules. Modules are used to divide the large projects into parts. So that they are individually developed, tested and maintained.

• AngularJS supports URL Routing. So that the URL (address) can be dynamically changes every time when a view is changed in the web page.

- AngularJS supports unit testing. That means each small component of angularjs can be individually run, by sending a dummy request, and then we can make sure the component returns a correct value.

Most Important Concepts of AngularJS :

In AngularJS, the following concepts are very important:

1. Views
2. Directives
3. Models
4. Controllers
5. Modules
6. Scopes
7. Dependency Injection
8. Filters
9. AJAX
10. Routing
11. Unit Testing

MVC Pattern in AngularJS

The project code is divided as 3 major parts:

1. Model
2. View
3. Controller

Model: It represents data of the application. In AngularJS, "model" is a "JSON object" that stores data.

View: It represents the html tags to display model data. So "data bindings" are maintained between "model" and "view". When some changes happen in the "view", then automatically AngularJS changes the same in the "model". When some changes happen in the "model", then automatically AngularJS changes the same in the "view".

Controller: It represents the JavaScript function, to implement functionality. Here "functionality" means calculations, validations, AJAX calls etc.

Controller can manipulate the model and view.

AngularJS::

AngularJS extends HTML with new attributes.
AngularJS is perfect for Single Page Applications (SPAs).
AngularJS is easy to learn.

```
 <!DOCTYPE html>
<html lang="en-US">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/
angular.min.js"></script>
<body>

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

</body>
</html>
```

AngularJS Example

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

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

</body>
</html>
```

AngularJS starts automatically when the web page has loaded.

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

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

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

AngularJS Directives

As you have already seen, AngularJS directives are HTML attributes with an ng prefix.
The ng-init directive initializes AngularJS application variables.

AngularJS Example

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

Alternatively with valid HTML: AngularJS Example

```
<div data-ng-app="" data-ng-init="firstName='srinivas'">

<p>The name is <span data-ng-bind="firstName"></span></p>

</div>
```

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

AngularJS Expressions

AngularJS binds data to HTML using Expressions.

AngularJS expressions can be written inside double braces: {{ expression }}.

AngularJS expressions can also be written inside a directive: ng-bind="expression".

AngularJS will resolve the expression, and return the result exactly where the expression is written.

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

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

Example

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

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

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

AngularJS Modules

An AngularJS module defines an application.
The module is a container for the different parts of an application.
The module is a container for the application controllers.
Controllers always belong to a module.

Creating a Module
A module is created by using the AngularJS function angular.module

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

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

The "myApp" parameter refers to an HTML element in which the application will run.
Now you can add controllers, directives, filters, and more, to your AngularJS application.

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

Example

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

<script>

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

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

</script>
```

AngularJS Directives
AngularJS directives are extended HTML attributes with the prefix ng-.
The ng-app directive initializes an AngularJS application.
The ng-init directive initializes application data.
The ng-model directive binds the value of HTML controls (input, select,

textarea) to application data.
Example
```
<div ng-app="" ng-init="firstName='srinivas'">

<p>Name: <input type="text" ng-model="firstName"></p>
<p>You wrote: {{ firstName }}</p>

</div>
```

Repeating HTML Elements
The ng-repeat directive repeats an HTML element:
Example
```
<div ng-app="" ng-init="names=['srinivas','Suresh','Sridhar']">
  <ul>
    <li ng-repeat="x in names">
      {{ x }}
    </li>
  </ul>
</div>
```

The ng-model Directive

With the ng-model directive you can bind the value of an input field to a variable created in AngularJS.
Example
```
<div ng-app="myApp" ng-controller="myCtrl">
    Name: <input ng-model="name">
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.name = "Srinivas Gorantla";
});
</script>
```

AngularJS Data Binding

Data binding in AngularJS is the synchronization between the model and the view.
Data Model
AngularJS applications usually have a data model. The data model is a collection of data available for the application.
Example
```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstname = "srinivas";
    $scope.lastname = "gorantla";
});
```

HTML View

The HTML container where the AngularJS application is displayed, is called the view.
The view has access to the model, and there are several ways of displaying model data in the view.
You can use the ng-bind directive, which will bind the innerHTML of the element to the specified model property:
Example

```
<p ng-bind="firstname"></p>
```

You can also use double braces {{ }} to display content from the model:
Example

```
<p>First name: {{firstname}}</p>
```

Two-way Binding

Data binding in AngularJS is the synchronization between the model and the view.
When data in the model changes, the view reflects the change, and when data in the view changes, the model is updated as well. This happens immediately and automatically, which makes sure that the model and the view is updated at all times.
Example

```
<div ng-app="myApp" ng-controller="myCtrl">
    Name: <input ng-model="firstname">
    <h1>{{firstname}}</h1>
</div>

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


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

<div ng-app="myApp" ng-controller="myCtrl">
    <h1 ng-click="changeName()">{{firstname}}</h1>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstname = "srinivas";
    $scope.changeName = function() {
        $scope.firstname = "Sandeeep";
    }
});
</script>

<p>Click on the header to run the "changeName" function.</p>

<p>This example demonstrates how to use the controller to change model
data.</p>

</body>
</html>
```

AngularJS Controllers

AngularJS controllers control the data of AngularJS applications.
AngularJS controllers are regular JavaScript Objects.

AngularJS Controllers

AngularJS applications are controlled by controllers.
The ng-controller directive defines the application controller.
A controller is a JavaScript Object, created by a standard JavaScript object constructor.
AngularJS Example

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

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

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

Application explained:

The AngularJS application is defined by ng-app="myApp". The application runs inside the <div>.
The ng-controller="myCtrl" attribute is an AngularJS directive. It defines a controller.

The myCtrl function is a JavaScript function.
AngularJS will invoke the controller with a $scope object.
In AngularJS, $scope is the application object (the owner of application variables and functions).
The controller creates two properties (variables) in the scope (firstName and lastName).
The ng-model directives bind the input fields to the controller properties (firstName and lastName).

Controller Methods

The example above demonstrated a controller object with two properties: lastName and firstName.
A controller can also have methods (variables as functions):

AngularJS Example

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

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{fullName()}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('personCtrl', function($scope) {
    $scope.firstName = "srinivas";
    $scope.lastName = "Gorantla";
    $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
    };
});
</script>
```

Controllers In External Files

In larger applications, it is common to store controllers in external files.
Just copy the code between the <script> tags into an external file named personController.js
AngularJS Example

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

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{fullName()}}

</div>
```

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

Another Example

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

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

Save the file as namesController.js

And then use the controller file in an application:
AngularJS Example

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

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

</div>

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

AngularJS Scope

The scope is the binding part between the HTML (view) and the JavaScript (controller).
The scope is an object with the available properties and methods.
The scope is available for both the view and the controller.

How to Use the Scope?
When you make a controller in AngularJS, you pass the $scope object as an argument:

Example:

Properties made in the controller, can be referred to in the view:

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

<h1>{{carname}}</h1>

</div>
```

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

app.controller('myCtrl', function($scope) {
    $scope.carname = "Volvo";
});
</script>
```

When adding properties to the $scope object in the controller, the view (HTML) gets access to these properties.
In the view, you do not use the prefix $scope, you just refer to a propertyname, like {{carname}}.

Understanding the Scope

If we consider an AngularJS application to consist of:

- View, which is the HTML.
- Model, which is the data available for the current view.
- Controller, which is the JavaScript function that makes/changes/removes/controls the data.

Then the scope is the Model.
The scope is a JavaScript object with properties and methods, which are available for both the view and the controller.
Example
If you make changes in the view, the model and the controller will be updated:

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

<input ng-model="name">

<h1>My name is {{name}}</h1>

</div>

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

app.controller('myCtrl', function($scope) {
    $scope.name = "Srinivas Gorantla";
});
</script>
```

Know Your Scope

It is important to know which scope you are dealing with, at any time.
In the two examples above there is only one scope, so knowing your scope is not an issue, but for larger applications there can be sections in the HTML DOM which can only access certain scopes.
Example
When dealing with the ng-repeat directive, each repetition has access to the current repetition object:

```
<div ng-app="myApp" ng-controller="myCtrl">
<ul>
    <li ng-repeat="x in names">{{x}}</li>
</ul>

</div>

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

app.controller('myCtrl', function($scope) {
    $scope.names = ["iPhone", "iPad", "iPod"];
});
</script>
```

Each <li> element has access to the current repetition object, in this case a string, which is referred to by using x.

Root Scope

All applications have a $rootScope which is the scope created on the HTML element that contains the ng-app directive.
The rootScope is available in the entire application.
If a variable has the same name in both the current scope and in the rootScope, the application use the one in the current scope.
Example
A variable named "color" exists in both the controller's scope and in the rootScope:

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

<p>The rootScope's favorite color:</p>
<h1>{{color}}</h1>

<div ng-controller="myCtrl">
    <p>The scope of the controller's favorite color:</p>
    <h1>{{color}}</h1>
</div>

<p>The rootScope's favorite color is still:</p>
<h1>{{color}}</h1>

<script>
var app = angular.module('myApp', []);
app.run(function($rootScope) {
    $rootScope.color = 'blue';
});
app.controller('myCtrl', function($scope) {
    $scope.color = "red";
});
</script>
```

```
</body>
```

AngularJS Filters

Filters can be added in AngularJS to format data.
AngularJS provides filters to transform data:

- currency Format a number to a currency format.
- date Format a date to a specified format.
- filter Select a subset of items from an array.
- json Format an object to a JSON string.
- limitTo Limits an array/string, into a specified number of elements/ characters.
- lowercase Format a string to lower case.
- number Format a number to a string.
- orderBy Orders an array by an expression.
- uppercase Format a string to upper case.

Adding Filters to Expressions

Filters can be added to expressions by using the pipe character |, followed by a filter.
The uppercase filter format strings to upper case:
Example

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

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

<script>
angular.module('myApp', []).controller('personCtrl', function($scope) {
    $scope.firstName = "srinivas",
    $scope.lastName = "Gorantla"
});
</script>

</body>
</html>
```

Example

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

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

<p>The name is {{ lastName | lowercase }}</p>

</div>

<script>
angular.module('myApp', []).controller('personCtrl', function($scope) {
    $scope.firstName = "Srinivas",
    $scope.lastName = "Gorantla"
});
</script>

</body>
</html>
```

Adding Filters to Directives

Filters are added to directives, like ng-repeat, by using the pipe character |, followed by a filter:
Example
The orderBy filter sorts an array:

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

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

</div>
```

The currency Filter

The currency filter formats a number as currency:
Example

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

<h1>Price: {{ price | currency }}</h1>

</div>
```

The filter Filter

The filter filter selects a subset of an array.
The filter filter can only be used on arrays, and it returns an array containing only the matching items.

Example

Return the names that contains the letter "i":
```
<div ng-app="myApp" ng-controller="namesCtrl">

<ul>
  <li ng-repeat="x in names | filter : 'i'">
    {{ x }}
  </li>
</ul>

</div>
```

Custom Filters

You can make your own filters by registering a new filter factory function with your module:

Example

Make a custom filter called "myFormat":
```
<ul ng-app="myApp" ng-controller="namesCtrl">
    <li ng-repeat="x in names">
        {{x | myFormat}}
    </li>
</ul>

<script>
var app = angular.module('myApp', []);
app.filter('myFormat', function() {
    return function(x) {

        var i, c, txt = "";
        for (i = 0; i < x.length; i++) {
            c = x[i];
            if (i % 2 == 0) {
                c = c.toUpperCase();
            }
            txt += c;
        }
        return txt;
    };
});

app.controller('namesCtrl', function($scope) {
    $scope.names = ['Jani', 'Carl', 'Margareth', 'Hege', 'Joe', 'Gustav', 'Birgit',
'Mary', 'Kai'];
});
</script>
```
The myFormat filter will format every other character to uppercase.

Example

```
<html>

  <head>
    <title>Angular JS Filters</title>
    <script type="text/javascript" src="js/angular.js" ></script>
  </head>
  <body>
    <h2>AngularJS Filters Application</h2>
    <div ng-app = "mainApp" ng-controller = "studentController">

      <table border = "0">
        <tr>
          <td>Enter first name:</td>
          <td><input type = "text" ng-model = "student.firstName"></td>
        </tr>

        <tr>
          <td>Enter last name: </td>
          <td><input type = "text" ng-model = "student.lastName"></td>
        </tr>
        <tr>
          <td>Enter fees: </td>
          <td><input type = "text" ng-model = "student.fees"></td>
        </tr>

        <tr>
          <td>Enter subject: </td>
          <td><input type = "text" ng-model = "subjectName"></td>
        </tr>
      </table>
      <br/>

      <table border = "0">
        <tr>
          <td>Name in Upper Case: </td><td>{{student.fullName() |
uppercase}}</td>
        </tr>

        <tr>
          <td>Name in Lower Case: </td><td>{{student.fullName() |
lowercase}}</td>
        </tr>

        <tr>
          <td>fees: </td><td>{{student.fees | currency:"&#8377;"}}
          </td>
        </tr>
```

```
            <tr>
               <td>Subject:</td>

               <td>
                  <ul>
                     <li ng-repeat = "subject in student.subjects | filter:
subjectName | orderBy:'name'">
                        {{ subject.name + ', marks:' + subject.marks }}
                     </li>
                  </ul>
               </td>
            </tr>
         </table>
      </div>

      <script>
       // angular module
         var mainApp = angular.module("mainApp", []);
         // angular controller
         mainApp.controller('studentController', function($scope) {

            $scope.student = {
               firstName: "Srinivas",
               lastName: "Gorantla",
               fees:500,

               subjects:[
                  {name:'Physics',marks:70},
                  {name:'Chemistry',marks:80},
                  {name:'Math',marks:65}
               ],

               fullName: function() {
                  var studentObject;
                  studentObject = $scope.student;
                  return studentObject.firstName + " " + studentObject.lastName;
               }
            };
         });
      </script>

   </body>
</html>
```

Tables :

```
<html>
   <head>
```

```
  <title>Angular JS Table</title>
  <script type="text/javascript" src="js/angular.js" ></script>

  <style>
  table{
    width: 1000px;
    color: red;
  }
  table tr td{
    width: 400px;
  }
    table, th , td {
      border: 1px solid grey;
      border-collapse: collapse;
      padding: 5px;
    }

    table tr:nth-child(odd) {
      background-color: #f2f2f2;
    }

    table tr:nth-child(even) {
      background-color: #ffffff;
    }
  </style>

</head>
<body>
  <h2>AngularJS Table Application</h2>
  <div ng-app = "mainApp" ng-controller = "studentController">

    <table border = "0">
      <tr>
        <td>Enter first name:</td>
        <td><input type = "text" ng-model = "student.firstName"></td>
      </tr>

      <tr>
        <td>Enter last name: </td>
        <td>
          <input type = "text" ng-model = "student.lastName">
        </td>
      </tr>

      <tr>
        <td>Name: </td>
        <td>{{student.fullName()}}</td>
      </tr>

      <tr>
        <td>Subject:</td>

        <td>
          <table>
            <tr>
              <th>Name</th>.
              <th>Marks</th>
            </tr>

            <tr ng-repeat = "subject in student.subjects">
              <td>{{ subject.name }}</td>
              <td>{{ subject.marks }}</td>
            </tr>

          </table>
        </td>

      </tr>
    </table>

  </div>

  <script>

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

    mainApp.controller('studentController', function($scope) {

      $scope.student = {

        firstName: "Srinivas",
        lastName: "Gorantla",
        fees:500,

        subjects:[

          {name:'Physics',marks:70},
          {name:'Chemistry',marks:80},
          {name:'Math',marks:65},
          {name:'English',marks:75},
          {name:'Hindi',marks:67}
        ],

        fullName: function() {
          var studentObject;
          studentObject = $scope.student;
          return studentObject.firstName + " " + studentObject.lastName;
        }
      };
    });
```

```
      </script>

   </body>
</html>
```

AngularJS HTML DOM

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

The ng-disabled Directive

The ng-disabled directive binds AngularJS application data to the disabled attribute of HTML elements.
The ng-disabled directive binds the application data mySwitch to the HTML button's disabled attribute.
The ng-model directive binds the value of the HTML checkbox element to the value of mySwitch.
If the value of mySwitch evaluates to true, the button will be disabled:
If the value of mySwitch evaluates to false, the button will not be disabled:

The ng-show Directive

The ng-show directive shows or hides an HTML element.
The ng-show directive shows (or hides) an HTML element based on the value of ng-show.
You can use any expression that evaluates to true or false:
The ng-hide Directive
The ng-hide directive hides or shows an HTML element.

```
<html>
   <head>
      <title>AngularJS HTML DOM</title>
      <style type="text/css">
body
{
   color: red;
   font-family: tahoma;
   font-size: 20px;
}
      </style>
   </head>

   <body>
      <h2>AngularJS DOM Application</h2>
      <div ng-app = "">

         <table border = "0">
            <tr>
            <td><input type = "checkbox" ng-model =
"enableDisableButton">Disable Button</td>
            <td><button ng-disabled = "enableDisableButton">Click Me!</
button></td>
          </tr>

          <tr>
            <td><input type = "checkbox" ng-model = "sd">Show Button</td>
            <td><button ng-show = "sd">Click Me!</button></td>
          </tr>

          <tr>
            <td><input type = "checkbox" ng-model = "showHide2">Hide
Button</td>
            <td><button ng-hide = "showHide2">Click Me!</button></td>
          </tr>

          <tr>
            <td><p>Total click: {{ clickCounter }}</p></td>
            <td><button ng-click = "clickCounter = clickCounter + 1">Click Me!
</button></td>
          </tr>
        </table>

      </div>

      <script type="text/javascript" src="js/angular.js" ></script>

   </body>
</html>
```

AngularJS Forms

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

Input Controls

Input controls are the HTML input elements:

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

```
<html>
   <head>
      <title>Angular JS Forms</title>
      <script type="text/javascript" src="js/angular.js" ></script>

      <style>
        table, th , td {
          border: 1px solid grey;
          border-collapse: collapse;
          padding: 5px;
        }

        table tr:nth-child(odd) {
          background-color: #f2f2f2;
        }
        table tr:nth-child(even) {
          background-color: #ffffff;
        }
      </style>

   </head>
   <body>

      <h2>AngularJS Form Application</h2>

      <div ng-app = "mainApp" ng-controller = "studentController">

        <form name = "studentForm" novalidate>
          <table border = "0">
            <tr>
              <td>Enter first name:</td>
              <td><input name = "firstname" type = "text" ng-model =
"firstName" required>
                  <span style = "color:red" ng-show = "studentForm.firstname.
$dirty && studentForm.firstname.$invalid">
                  <span ng-show = "studentForm.firstname.
$error.required">First Name is required.</span>
                </span>
              </td>
            </tr>

            <tr>
              <td>Enter last name: </td>
              <td><input name = "lastname"  type = "text" ng-model =
"lastName" required>
                <span style = "color:red" ng-show = "studentForm.lastname.
$dirty && studentForm.lastname.$invalid">
                  <span ng-show = "studentForm.lastname.
$error.required">Last Name is required.</span>
                </span>
              </td>
            </tr>

            <tr>
              <td>Email: </td><td><input name = "email" type = "email" ng-
model = "email" length = "100" required>
                <span style = "color:red" ng-show = "studentForm.email.$dirty
```

```
&& studentForm.email.$invalid">
                    <span ng-show = "studentForm.email.$error.required">Email
is required.</span>
                    <span ng-show = "studentForm.email.$error.email">Invalid
email address.</span>
                  </span>
                </td>
              </tr>

              <tr>
                <td>
                  <button ng-click = "reset()">Reset</button>
                </td>
                <td>
                  <button ng-disabled = "studentForm.firstname.$dirty &&
                    studentForm.firstname.$invalid || studentForm.lastname.$dirty
&&
                    studentForm.lastname.$invalid || studentForm.email.$dirty &&
                    studentForm.email.$invalid" ng-click="submit()">Submit</
button>
                </td>
              </tr>

            </table>
          </form>
        </div>

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

          mainApp.controller('studentController', function($scope) {

            $scope.reset = function(){
              $scope.firstName = "Srinivas";
              $scope.lastName = "Gorantla";
              $scope.email = "gsrinu004@gmail.com";
            }

            $scope.reset();
          });
        </script>

      </body>
    </html>
```

AngularJS Includes

With AngularJS, you can include HTML from an external file.

AngularJS Includes

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

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

<div ng-include="'myFile.htm'"></div>

</body>
```

Example

```
<html>
  <head>
    <title>Angular JS Includes</title>
    <script type="text/javascript" src="js/angular.js" ></script>

    <style>
      table, th , td {
        border: 1px solid grey;
        border-collapse: collapse;
        padding: 5px;
      }

      table tr:nth-child(odd) {
        background-color: #f2f2f2;
      }

      table tr:nth-child(even) {
        background-color: #ffffff;
      }
    </style>

  </head>
  <body>

    <h2>AngularJS Include Application</h2>

    <div ng-app = "mainApp" ng-controller="studentController">

      <div ng-include = "'include/main.htm'"></div>
      <div ng-include = "'include/subjects.htm'"></div>
    </div>

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

      mainApp.controller('studentController', function($scope) {
        $scope.student = {
          firstName: "Srinivas",
          lastName: "Gorantla",
          fees:500,

          subjects:[
             {name:'Physics',marks:70},
             {name:'Chemistry',marks:80},
             {name:'Math',marks:65},
             {name:'English',marks:75},
             {name:'Hindi',marks:67}
          ],

          fullName: function() {
             var studentObject;
             studentObject = $scope.student;
             return studentObject.firstName + " " + studentObject.lastName;
          }
        };
      });
    </script>

  </body>
</html>
```

AngularJS AJAX - $http

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

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

Example

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

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

<p>Today's welcome message is:</p>
<h1>{{myWelcome}}</h1>

</div>

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

Methods

The example above uses the .get method of the $http service.
The .get method is a shortcut method of the $http service. There are several shortcut methods:
.delete()
.get()
.head()
.jsonp()
.patch()
.post()
.put()

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

Example

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

The example above executes the $http service with an object as an argument. The object is specifying the HTTP method, the url, what to do on success, and what to do on failure.
Properties
The response from the server is an object with these properties:
.config the object used to generate the request.
.data a string, or an object, carrying the response from the server.
.headers a function to use to get header information.
.status a number defining the HTTP status.
.statusText a string defining the HTTP status.

```
<html>
   <head>
      <title>Angular JS Ajax</title>
      <style>
         table, th , td {
            border: 1px solid grey;
            border-collapse: collapse;
            padding: 5px;
         }
         table tr:nth-child(odd) {
            background-color: #f2f2f2;
         }
         table tr:nth-child(even) {
            background-color: #ffffff;
         }
      </style>
   </head>
   <body>
      <h2>AngularJS Sample Application</h2>
      <div ng-app = "myApp" ng-controller = "studentController">
         <table>
            <tr>
               <th>Name</th>
               <th>Roll No</th>
               <th>Percentage</th>
            </tr>
            <tr ng-repeat = "student in StuRecords">
               <td>{{ student.Name }}</td>
               <td>{{ student.RollNo }}</td>
               <td>{{ student.Percentage }}</td>
            </tr>
         </table>
      </div>
<script type="text/javascript" src="js/angular.js"></script>
       <script>
 var app = angular.module('myApp', []);
app.controller('studentController', function($scope, $http) {
  $http.get("data.json").then(function (response) {
      $scope.StuRecords = response.data;
  });
});
      </script>
   </body>
</html>
```

Data.json:

```
[
   {
      "Name" : "Srinivas Gorantla",
      "RollNo" : 101,
      "Percentage" : "80%"
   },
   {
      "Name" : "Suresh Kata",
      "RollNo" : 102,
      "Percentage" : "70%"
   },
   {
      "Name" : "Mahesh Boyna",
      "RollNo" : 103,
    "Percentage" : "75%"
  },
  {
    "Name" : "Kalyan Kurra",
    "RollNo" : 104,
    "Percentage" : "77%"
  }
]
```

AngularJS Events

AngularJS has its own HTML events directives.

AngularJS Events

You can add AngularJS event listeners to your HTML elements by using one or more of these directives:

- ng-blur
- ng-change
- ng-click
- ng-copy
- ng-cut
- ng-dblclick
- ng-focus
- ng-keydown
- ng-keypress
- ng-keyup
- ng-mousedown
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-mouseup
- ng-paste

The event directives allows us to run AngularJS functions at certain user events.
An AngularJS event will not overwrite an HTML event, both events will be executed.

Mouse Events

Mouse events occur when the cursor moves over an element, in this order:

1. ng-mouseenter
2. ng-mouseover
3. ng-mousemove
4. ng-mouseleave

Or when a mouse button is clicked on an element, in this order:

1. ng-mousedown
2. ng-mouseup

3. ng-click

You can add mouse events on any HTML element.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/
angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<h1 ng-mousemove="count = count + 1">Mouse Over Me!</h1>
<h2>{{ count }}</h2>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
   $scope.count = 0;
});
</script>
</body>
</html>
```

The ng-click Directive

The ng-click directive defines AngularJS code that will be executed when the element is being clicked.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/
angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<button ng-click="count = count + 1">Click Me!</button>
<p>{{ count }}</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
   $scope.count = 0;
});
</script>
</body>
</html>
```

Example -2::

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

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

Example -3

Toggle, True/False
If you want to show a section of HTML code when a button is clicked, and hide when the button is clicked again, like a dropdown menu, make the button behave like a toggle switch:

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/
angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<button ng-click="myFunc()">Click Me!</button>
<div ng-show="showMe">
    <h1>Menu:</h1>
    <div>Pizza</div>
    <div>Pasta</div>
    <div>Pesce</div>
</div>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.showMe = false;
    $scope.myFunc = function() {
        $scope.showMe = !$scope.showMe;
    }
});
</script>
<p>Click the button to show/hide the menu.</p>
</body>
</html>
```

$event Object

You can pass the $event object as an argument when calling the function.
The $event object contains the browser's event object:

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/
angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<h1 ng-mousemove="myFunc($event)">Mouse Over Me!</h1>
<p>Coordinates: {{x + ', ' + y}}</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.myFunc = function(myE) {
        $scope.x = myE.clientX;
        $scope.y = myE.clientY;
    }
});
</script>
<p>Mouse over the heading to display the value of clientX and clientY from the
event object.</p>
</body>
</html>
```

AngularJS Animations

AngularJS provides animated transitions, with help from CSS.
What is an Animation?
An animation is when the transformation of an HTML element gives you an illusion of motion.

```
<!DOCTYPE html>
<html>
<style>
div {
  transition: all linear 0.5s;
  background-color: lightblue;
  height: 100px;
  width: 100%;
  position: relative;
  top: 0;
  left: 0;
}
.ng-hide {
  height: 0;
  width: 0;
  background-color: transparent;
  top:-200px;
  left: 200px;
}
</style>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/
angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-
animate.js"></script>
<body ng-app="ngAnimate">
<h1>Hide the DIV: <input type="checkbox" ng-model="myCheck"></h1>
<div ng-hide="myCheck"></div>
</body>
</html>
```

AngularJS Routing

The ngRoute module helps your application to become a Single Page Application.
What is Routing in AngularJS?

If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.

The ngRoute module routes your application to different pages without reloading the entire application.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/
angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular-
route.js"></script>

<body ng-app="myApp">

<p><a href="#/!">Main</a></p>

<a href="#!red">Red</a>
<a href="#!green">Green</a>
<a href="#!blue">Blue</a>

<div ng-view></div>

<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.htm"
    })
    .when("/red", {
        templateUrl : "red.htm"
    })
    .when("/green", {
        templateUrl : "green.htm"
    })
    .when("/blue", {
        templateUrl : "blue.htm"
    });
});
</script>

<p>Click on the links to navigate to "red.htm", "green.htm", "blue.htm", or
back to "main.htm"</p>
</body>
</html>
```

To make your applications ready for routing, you must include the AngularJS Route module:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular-
route.js"></script>
```

Then you must add the ngRoute as a dependency in the application module::

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

Controllers

With the $routeProvider you can also define a controller for each "view".

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/
angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular-
route.js"></script>

<body ng-app="myApp">

<p><a href="#/!">Main</a></p>

<a href="#!hyderabad">Hyderabad</a>
<a href="#!chennai">Chennai</a>

<p>Click on the links.</p>

<p>Note that each "view" has its own controller which each gives the "msg"
variable a value.</p>

<div ng-view></div>
```

```
<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider

    .when("/", {
        templateUrl : "main.html",
    })
    .when("/hyderabad", {
        templateUrl : "hyd.htm",
        controller : "HydCtrl"
    })
    .when("/chennai", {
        templateUrl : "chennai.htm",
        controller : "ChennaiCtrl"
    });
});
app.controller("HydCtrl", function ($scope) {
    $scope.msg = "I love Hyderabad";
});
app.controller("ChennaiCtrl", function ($scope) {
    $scope.msg = "I love Chennai";
});

</script>

</body>
</html>
```

chennai.htm

```
<h1>Chennai</h1>
<h3>Chennai is the one of the city of India.</h3>
<p>{{msg}}</p>
```

hyd.htm

```
<h1>Hyderabad</h1>
<h3>Hyderabad is the one of the city of India.</h3>
<p>{{msg}}</p>
```

Adding multiple modules for HTML

Example

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

```
    <head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/
angular.min.js"></script>
        <script src="angular.ng-modules.js"></script>

        <script>
          var moduleA = angular.module("MyModuleA", []);
          moduleA.controller("MyControllerA", function($scope) {
              $scope.name = "Srinivas";
          });
          var moduleB = angular.module("MyModuleB", []);
          moduleB.controller("MyControllerB", function($scope) {
              $scope.name = "Santhosh";
          });
        </script>

    </head>
    <body>
        <div ng-modules="MyModuleA, MyModuleB">
            <h1>Module A, B</h1>
            <div ng-controller="MyControllerA">
                {{name}}
            </div>
            <div ng-controller="MyControllerB">
                {{name}}
            </div>
        </div>

        <div ng-module="MyModuleB">
            <h1>Just Module B</h1>
            <div ng-controller="MyControllerB">
                {{name}}
            </div>
        </div>
    </body>
</html>
```

Example

```
<!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-modules="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>
```

AngularController.js

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

firstModule.controller('firstController', function ($scope) {
    $scope.UserName = 'Srinivas Gorantla';
});

firstModule.controller('secondController', function ($scope) {
    $scope.MobileNumber = '9390XXX007';
});

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

secondModule.controller('thirdController', function ($scope) {
    $scope.EmailId = 'gsrinu004@gmail.com';
});
```