AngularJS is a client side, open source JavaScript MVC framework for web application or web sites. It extends the HTML and makes it dynamic. AngularJS can be used to create Single Page Applications.

AngularJS is entirely based on HTML and JavaScript, so there is no need to learn another syntax or language.

AngularJS changes static HTML to dynamic HTML. It extends the ability of HTML by adding built-in attributes and components and also provides an ability to create custom attributes using simple JavaScript.

AngularJS

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/1.6.9/angular.min.js"></script>
```

AngularJS Extends HTML

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

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

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

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

AngularJS Example

```
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/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>
```

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

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

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

Setup AngularJS Development Environment:

We need the following tools to setup a development environment for AngularJS:

1. AngularJS Library
2. Editor/IDE
3. Browser
4. Web server

AngularJS Library:

To download AngularJS library, go to angularjs.org ->

Editor:

AngularJS is eventually HTML and JavaScript code. So you can install any good editor/IDE as per your choice.

The following editors are recommended:

- Sublime Text
- Aptana Studio 3
- Ultra Edit
- Eclipse
- Visual Studio

Online Editor:

You can also use the following online editors for learning purpose.

- plnkr.co
- jsbin.com

  We are using our own online code editor for all the AngularJS examples in these tutorials.

Web server:

Use any web server such as IIS, apache etc., locally for development purpose.

Browser:

You can install any browser of your choice as AngularJS supports cross-browser compatibility. However, it is recommended to use Google Chrome while developing an application.

AngularJS Directives

AngularJS directives are HTML attributes with an **ng** prefix.

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.

The ng-app Directive

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

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

The ng-init Directive

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

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

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

Input something in the input box:

Name: <input type="text" ng-model="firstName">

You wrote: {{ firstName }}

</body>

</html>

The ng-model Directive

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

The ng-model directive can also:

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

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

Example

```
</script> <!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

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

Name: <input ng-model="name">

</div>

<script>

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

app.controller('myCtrl', function($scope) {
```

```
    $scope.name = "John Doe";
});
</script>
<p>Use the ng-model directive to bind the value of the input field to a property made in the controller.</p>
</body>
</html>
```

Repeating HTML Elements

The ng-repeat directive repeats an HTML element:

The ng-repeat directive actually clones HTML elements **once for each item in a collection.**

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

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="" ng-init="names=[{name:'Jani',country:'Norway'},
{name:'Hege',country:'Sweden'},
{name:'Kai',country:'Denmark'}]">
<p>Looping with objects:</p>
<ul>
  <li ng-repeat="x in names">
  {{ x.name + ', ' + x.country }}</li>
</ul>
```

```
</div>


</body>

</html>
```

AngularJS 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 }}

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

AngularJS Example

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

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

</body>
</html>
```

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

AngularJS Example

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
```

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

</body>
</html>
```

Using ng-init is not very common. You will learn a better way to initialize data

```
<div ng-app="" ng-init="quantity=1;cost=5">

<p>Total in dollar: <span ng-bind="quantity * cost"></span></p>
</div>
```

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

AngularJS Strings

AngularJS strings are like JavaScript strings:

Example

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>The name is {{ firstName + " " + lastName }}</p>

</div>
```

AngularJS Objects

AngularJS objects are like JavaScript objects:

Example

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>The name is {{ person.lastName }}</p>

</div>
```

AngularJS Example

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

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

</div>

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

AngularJS Modules

An AngularJS module defines an application.

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

The module is a container for the application controllers.

Controllers always belong to a module.

Creating a Module

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

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

<script>

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

</script>
```

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

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

Adding a Controller

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

Example

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

</script>
```

```
<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/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 = "John";
```

```
    $scope.changeName = function() {
        $scope.firstname = "Nelly";
    }
});
</script>
```

Example

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<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>
</body>
</html>
```

program explained:

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

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

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

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

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

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

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

AngularJS Services

In AngularJS you can make your own service, or use one of the many built-in services.

What is a Service?

In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application.

AngularJS has about 30 built-in services. One of them is the $location service.

The $location service has methods which return information about the location of the current web page:

Example

Use the $location service in a controller:

```
<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

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

```
<p>The url of this page is:</p>

<h3>{{myUrl}}</h3>

</div><p>This example uses the built-in $location service to get the absolute url of the page.</p>

<script>

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

app.controller('myCtrl', function($scope, $location) {

   $scope.myUrl = $location.absUrl();

});

</script>

</body>

</html>
```

Note that the $location service is passed in to the controller as an argument. In order to use the service in the controller, it must be defined as a dependency.

Why use Services?

For many services, like the $location service, it seems like you could use objects that are already in the DOM, like the window.location object, and you could, but it would have some limitations, at least for your AngularJS application.

AngularJS constantly supervises your application, and for it to handle changes and events properly, AngularJS prefers that you use the $location service instead of the window.location object.

The $http Service

The $http service is one of the most common used services in AngularJS applications. The service makes a request to the server, and lets your application handle the response.

Example

Use the $http service to request data from the server:

```
<!DOCTYPE html>
```

```
<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

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

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

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

</div>

<p>The $http service requests a page on the server, and the response is set as the value of the "myWelcome" variable.</p>

<script>

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

app.controller('myCtrl', function($scope, $http) {

  $http.get("welcome.htm").then(function (response) {

      $scope.myWelcome = response.data;

  });

});

</script>

</body>

</html>
```

The $timeout Service

The `$timeout` service is AngularJS' version of the `window.setTimeout` function.

Example

Display a new message after two seconds:

```
<!DOCTYPE html>
```

```
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<p>This header will change after two seconds:</p>
<h1>{{myHeader}}</h1>
</div>
<p>The $timeout service runs a function after a specified number of milliseconds.</p>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $timeout) {
  $scope.myHeader = "Hello World!";
  $timeout(function () {
      $scope.myHeader = "How are you today?";
  }, 2000);
});
</script>
</body>
</html>
```

The $interval Service

The `$interval` service is AngularJS' version of the `window.setInterval` function.

Example

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

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

The time is:

h1>{{theTime}}</h1>

The $interval service runs a function every specified millisecond.

<script>

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

app.controller('myCtrl', function($scope, $interval) {

$scope.theTime = new Date().toLocaleTimeString();

$interval(function () {

$scope.theTime = new Date().toLocaleTimeString();

}, 2000);

});

</script>

</body>

</html>

Create Your Own Service

To create your own service, connect your service to the module:

Create a service named hexafy:

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

});

To use your custom made service, add it as a dependency when defining the controller:

Example

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<p>The hexadecimal value of 255 is:</p>
<h1>{{hex}}</h1>
</div>
<p>A custom service with a method that converts a given number into a hexadecimal number.</p>
<script>
var app = angular.module('myApp', []);
app.service('hexafy', function() {
    this.myFunc = function (x) {
        return x.toString(16);
    }
});
app.controller('myCtrl', function($scope, hexafy) {
  $scope.hex = hexafy.myFunc(255);
});
</script>
</body>
</html>
```