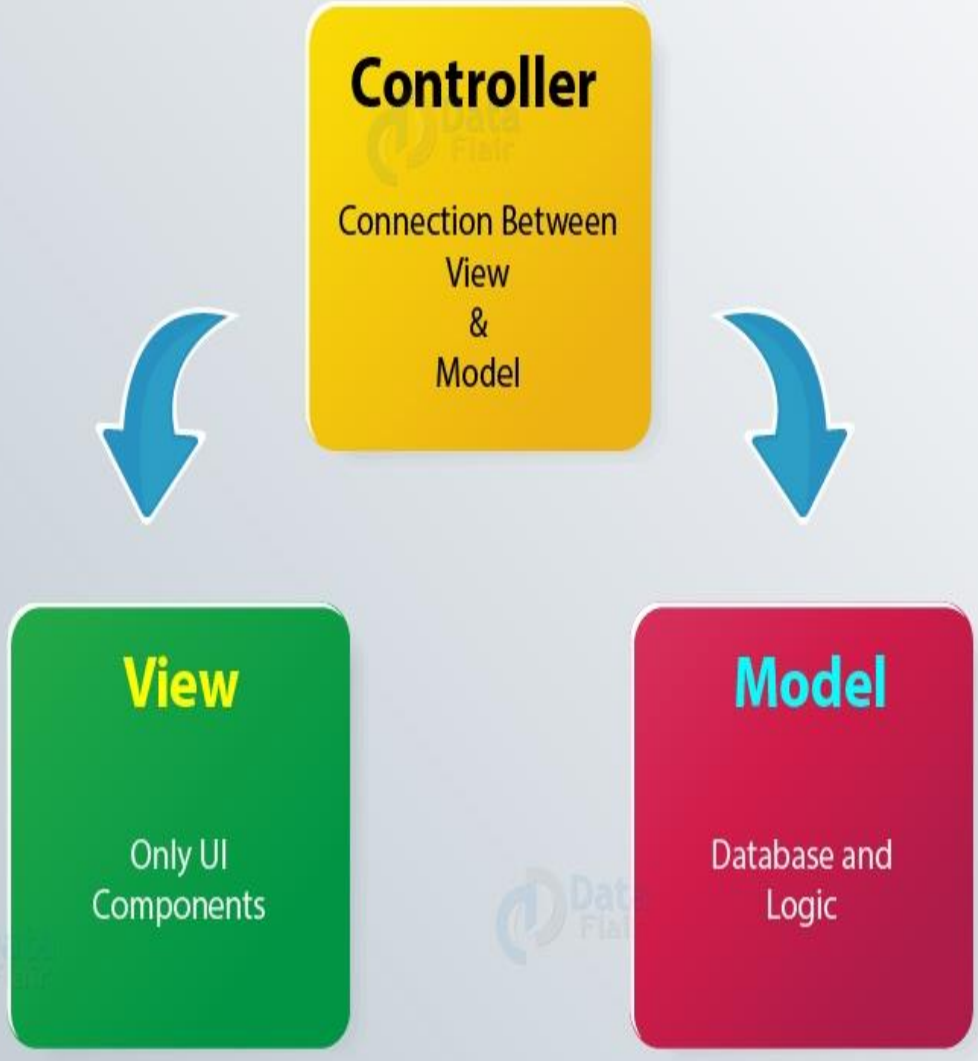
AngularJS

What is AngularJS?

- AngularJS is a client side JavaScript MVC framework to develop a dynamic web application.
- AngularJS was originally started as a project in Google but now, it is open source framework.
- AngularJS is entirely based on HTML and JavaScript, so there is no need to learn another syntax or language.
- AngularJS extends HTML with new attributes.

- **History of Angular Versions**
- Following are the Angular version release dates:
- Angular version 1.0 which is known as AngularJS was released in 2010 by Google
- Angular version 2.0 was released in September 2016
- Angular 4.0 was released in March 2017
- Angular 5.0 was released in Nov 2017
- Angular 6.0 was released in May 2018
- Angular 7.0 was released in Oct 2018
- Angular 8.0 was released in May 2019
- Angular 9.0 was released in Feb 2020
- Angular 10.0 was released in June 2020
- Angular 11.0 was released in Nov 2020

MVC Architecture (Basic)



- AngularJS is an open source JavaScript MVC framework for web application or web sites
- AngularJS can be used to create Single Page Applications.
- Prerequisites
- Basic knowledge of HTML, JavaScript, CSS and web application is required.

- 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.

What are the core features in AngularJS?

- Following are the core features of AngularJS –
- **Data-binding** — An automatic synchronization of data between model and view components.
- **Scope** — “Objects that refer to the model”. It acts as a glue between controller and view.
- **Controller** — “JavaScript functions that are bound to a particular scope”.
- **Services** – Several built-in services are there in AngularJS, for example, $https: to make a XMLHttpRequests.
- **Filters** – These select a subset of items from an array and return a new array.

- **Directives** – These are markers on DOM elements such as elements, attributes, CSS, and more. They are used to create custom HTML tags that serve the purpose of a new, custom widgets. AngularJS has built-in directives (ngBind, ngModel…)
- **Templates** — It is the rendered view with information from the controller and model. Using “partials”, there can be a single file (like index.html) or multiple views all in one page.
- **Routing** – It is a concept of switching views.
- **Model View Whatever** – Model View Whatever is inferred by the Angular JS team humorously because MVC can’t be implemented in the traditional sense in AngularJS, but can be in closer to Model-View-ViewModel (MVVM).
- **Deep Linking** — Encodes the state of an application in the URL for bookmarking. With the help of URL, the application can be restored back to the same state.
- **Dependency Injection** — With its built-in dependency injection subsystem, it makes it easy to develop, understand, and test the application.

AngularJS in Real World

Usage of AngularJS increases with these much plus points.

752,756

1.14 %

WEBSITES

312,213

UNIQUE DOMAINS

Top Website Categories

Website categories where Angular JS is being used

Website Category	Popularity
Business & Industry	18.53%
Internet & Telecom	8.60%
Arts & Entertainment	6.69%
Shopping	4.87%
Others	61.31%

- Setup AngularJS Development Environment
- We need the following tools to setup a development environment for AngularJS:

- AngularJS Library
- Editor/IDE
- Browser
- Web server

AngularJS Library

To download AngularJS library, go to angularjs.org -> click download butto which will open the following popup.

ages/ng/setup-env-1.png

Download AngularJS Library

- 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

- 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.

First AngularJS Application

- Let's create a simple AngularJS web application step by step and understand the basic building blocks of AngularJS.
- First, create an HTML document with <head> and <body> elements, as show below

- Example: HTML Template

- <!DOCTYPE html>
- <html>
- <head>
- </head>
- <body>
- </body>
- </html>

- Include angular.js file in the head section using 2 options.

Example: Include AngularJS Library

```
<!DOCTYPE html>
<html>
<head>
    <title>First AngularJS Application</title>
    <script src= "~/Scripts/angular.js"></script>
</head>
<body>

</body>
</html>
```

- Here, we will be creating a simple multiplier application which will multiply two numbers and display the result. User will enter two numbers in two separate textboxes and the result will be displayed immediately, as shown below.

- <!DOCTYPE html>

- <html>
- <head>
- <title>First AngularJS Application</title>
- <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
- </head>
- <body ng-app >
- <h1>First AngularJS Application</h1>

Enter Numbers to Multiply:

- <input type="text" ng-model="Num1" /> x <input type="text" ng-model="Num2" />
- = {{Num1 * Num2}}
- </body>
- </html>

Output:

Result:

First AngularJS Application

Enter Numbers to Multiply: 3 x 6 = 18

- The above example is looks like HTML code with some strange attributes and braces such as ng-app, ng-model, and {{ }}. These built-in attributes in AngularJS are called directives.
- The following figure illustrates the AngularJS building blocks in the above example.
-

```
<!DOCTYPE html>
<html>
<head>
    <title>First AngularJS Application</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app>
    <h1>First AngularJS Application</h1>
    Enter Numbers to Multiply:
    <input type="text" ng-model="Num1" /> x <input type="text" ng-model="Num2" />
    = <span>{{Num1 * Num2}}</span>
</body>
</html>
```

Template

Directives

Expression

- Template
- In AngularJS, a template is HTML with additional markups. AngularJS compiles templates and renders the resultant HTML.
- Directive
- Directives are markers (attributes) on a DOM element that tell AngularJS to attach a specific behavior to that DOM element or even transform the DOM element and its children.
- Most of the directives in AngularJS are starting with **ng**. It stands for Angular. We have applied ng-app and ng-model directive in the above example.

- **ng-app**: The ng-app directive is a starting point. If AngularJS framework finds ng-app directive anywhere in the HTML document then it bootstraps (initializes) itself and compiles the HTML template.
- **ng-model**: The ng-model directive binds HTML element to a property on the $scope object.

- Expression
- An expression is like JavaScript code which is usually wrapped inside double curly braces such as {{ expression }}. AngularJS framework evaluates the expression and produces a result. In the above example, {{ Num1 * Num2}} will simply display the product of Num1 and Num2.

- 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.

- The following table lists all the important concepts in AngularJS.

Concept	Description
Template	HTML with additional markup
Directives	Extends the HTML with custom attributes and elements
Model	The data shown to the user in the view and with which the user interacts
Scope	A context where the model is stored so that controllers, directives and expressions can access it
Expressions	Executes JavaScript code inside brackets {{ }}.
Compiler	Parses the template and instantiates directives and expressions
Filter	Formats the value of an expression for display to the user
View	what the user sees (the DOM)
Data Binding	Sync data between the model and the view
Controller	Maintains the application data and business logic
Module	a container for different parts of an app including controllers, services, filters, directives which configure the Injector

Scope	A context where the model is stored so that controllers, directives and expressions access it
Expressions	Executes JavaScript code inside brackets {{ }}.
Compiler	Parses the template and instantiates directives and expressions
Filter	Formats the value of an expression for display to the user
View	what the user sees (the DOM)
Data Binding	Sync data between the model and the view
Controller	Maintains the application data and business logic
Module	a container for different parts of an app including controllers, services, filters, direc which configure the Injector
Service	Reusable business logic, independent of views
Dependency Injection	Creates and wires objects and functions
Injector	Dependency injection container

The ng-app Directive

```
<!DOCTYPE html>
<html>
<head>
    <title>ng-app Directive</title>
    <script src="../Scripts/angular.min.js"></script>
</head>
<body >
    <div>
        {{2/2}}
    </div>
    <div id="myDiv" ng-app>
        {{5/2}}
        <div>
            {{10/2}}
        </div>
    </div>
    <div>
        {{2/2}}
    </div>
</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Angular Bootstrap</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body>
    <div>
        {{2/2}}
    </div>
    <div ng-app id="myDiv">
        {{5/2}}
        <div>
            {{10/2}}
        </div>
    </div>
    <div>
        {{2/2}}
    </div>
</body>
</html>
```


Angular features not supported out of ng-app

Angular features supported only inside ng-app

- Note : that multiple ng-app directives are **NOT** allowed in a single HTML document.

- Manual Bootstrap

- We have learned that the ng-app directive auto initializes an AngularJS framework. However, we can also initialize AngularJS manually without using ng-app directive.
- The following example demonstrates manual initialization of Angular.

Example: Manual Bootstrap

```
<!DOCTYPE html>
<html >
<head>
    <title>Angular Bootstrap</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body>
    <div>
         {{2/2}}
    </div>
    <div>
        {{5/2}}
        <div>
            {{10/2}}
        </div>
    </div>
<script>
     angular.element(document).ready(function () {
         angular.bootstrap(document);
     });
</script>
```

AngularJS Expression

- AngularJS expression is like JavaScript expression surrounded with braces - {{ expression }}. AngularJS evaluates the specified expression and binds the result data to HTML.

- AngularJS expression can contain literals, operators and variables like JavaScript expression. For example, an expression {{2/2}} will produce the result 1 and will be bound to HTML.

Example: Expression

```
<!DOCTYPE html>
<html >
<head>
    <script src="~/Scripts/angular.js"></script>
</head>
<body >
    <h1>AngularJS Expression Demo:</h1>
    <div ng-app>
        2 + 2 = {{2 + 2}} <br />
        2 - 2 = {{2 - 2}} <br />
        2 * 2 = {{2 * 2}} <br />
        2 / 2 = {{2 / 2}}
    </div>
</body>
</html>
```

Try it

- AngularJS expression is like JavaScript code expression except for the following differences:
- AngularJS expression cannot contain conditions, loops, exceptions or regular expressions e.g. if-else, ternary, for loop, while loop etc.
- AngularJS expression cannot declare functions.
- AngularJS expression cannot contain comma or void.
- AngularJS expression cannot contain return keyword.

Example: Expression

```
<html >
<head>
    <script src="~/Scripts/angular.js"></script>
</head>
<body >
    <h1>AngularJS Expression Demo:</h1>
    <div ng-app>
        {{"Hello World"}}<br />
        {{100}}<br />
        {{true}}<br />
        {{10.2}}
    </div>
</body>
</html>
```

Try it

Output:

- Hello World
- 100
- True
- 10.2

Example: Expression

```
<!DOCTYPE html>
<html >
<head>
    <script src="~/Scripts/angular.js"></script>
</head>
<body >
    <div ng-app>
        {{"Hello" + " World"}}<br />
        {{100 + 100 }}<br />
        {{true + false}}<br />
        {{10.2 + 10.2}}<br />
    </div>
</body>
</html>
```

Output:

- Result:

- Hello World
- 200
- 1
- 20.4

ng-init:

- The ng-init directive can be used to initialize variables in AngularJS application.
- The following example demonstrates ng-init directive that initializes variable of string, number, array, and object.

- Example: ng-init
- <!DOCTYPE html>
- <html >
- <head>
- <script src="~/Scripts/angular.js"></script> </head>
- <body >
- <div ng-app ng-init="greet='Hello World!'; amount= 100; myArr = [100, 200]; person = { firstName:'Steve', lastName :'Jobs'}">
- {{amount}}
- {{myArr[1]}}
- {{person.firstName}}
-
- </body>
- </html>

- Result:

- 100
- 200
- Steve

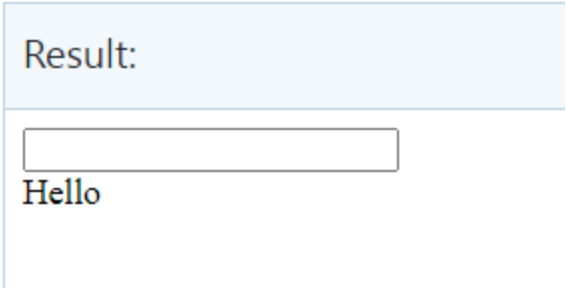
ng-model:

- The ng-model directive is used for two-way data binding in AngularJS. It binds <input>, <select> or <textarea> elements to a specified property on the $scope object.

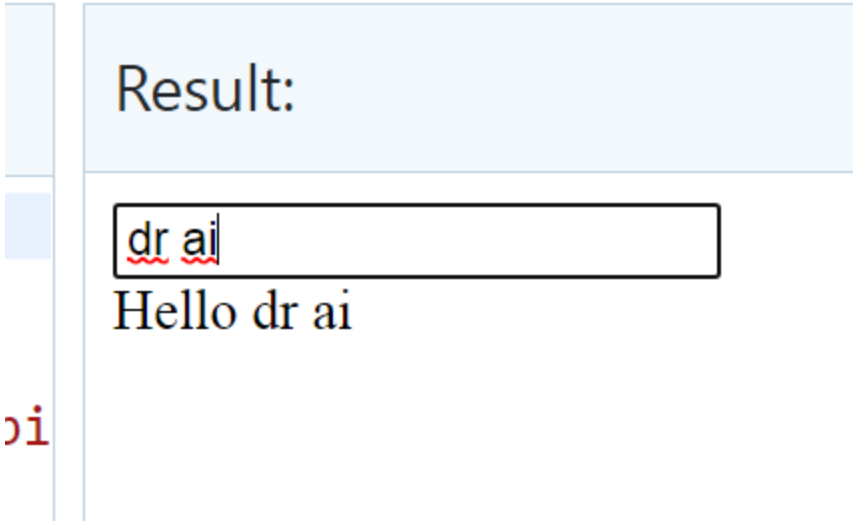
Example: ng-model

```html
<!DOCTYPE html>
<html >
<head>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app>
    <input type="text" ng-model="name" />
    <div>
       Hello {{name}}
    </div>
</body>
</html>
```

- output:



- After typing text



ng-bind

- !DOCTYPE html>
- <html >
- <head>
- <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
- </head>

- body ng-app="">
-
- 5 + 5 =

- Enter your name: <input type="text" ng-model="name" />
- Hello
-
- </body>
- </html>

Result:

5 + 5 = 10
Enter your name:
Hello

ng-init:

- The ng-init directive can be used to initialize variables in AngularJS application.
- The following example demonstrates ng-init directive that initializes variable of string, number, array, and object.

- <!DOCTYPE html>
- <html >
- <head>
- <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
- </head>
- <body >
- <div ng-app ng-init="greet='Hello World!'; amount= 100; myArr = [100, 200]; person = { firstName:'Steve', lastName :'Jobs'}">

- {{amount}}
- {{myArr[1]}}
- {{person.firstName}}
-
- </body>
- </html>
- Output:
- 100
  200
  Steve

AngularJS Controller

- The controller in AngularJS is a JavaScript function that maintains the application data and behavior using $scope object.
- ng-controller
- The ng-controller directive is used to specify a controller in HTML element, which will add behavior or maintain the data in that HTML element and its child elements.
- The following example demonstrates attaching properties to the $scope object inside a controller and then displaying property value in HTML.

Example: AngularJS Controller

```
<!DOCTYPE html>
<html >
<head>
    <title>AngualrJS Controller</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app="myNgApp">
    <div ng-controller="myController">
        {{message}}
    </div>
    <script>
        var ngApp = angular.module('myNgApp', []);

        ngApp.controller('myController', function ($scope) {
            $scope.message = "Hello World!";
        });
    </script>
</body>
</html>
```

Result:
Hello World!

```html
<!DOCTYPE html>
<html>
<head>
    <title>AngualrJS Controller</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app="myNgApp">
    <div ng-controller="myController">
        {{message}}
    </div>
    <script>
        var ngApp = angular.module('myNgApp', []);

        ngApp.controller('myController', function ($scope) {
            $scope.message = "Hello World!";
        });
    </script>
</body>
</html>
```


1. Specify a controller using ng-controller

2. Create an App module

3. Create a Controller

4. Attach a property to $scope

5. Use a property created inside a controller

Example: Controller

```
<!DOCTYPE html>
<html>
<head>
    <title>AngualrJS Controller</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app="myNgApp">
    <div id="div1" ng-controller="myController">
        Message: {{message}} <br />
        <div id="div2">
            Message: {{message}}
        </div>
    </div>
    <div id="div3">
        Message: {{message}}
    </div>
```

```
    <div id="div4" ng-controller="anotherController">
        Message: {{message}}
    </div>
    <script>
        var ngApp = angular.module('myNgApp', []);

        ngApp.controller('myController', function ($scope) {
            $scope.message = "This is myController";
        });

        ngApp.controller('anotherController', function ($scope) {
            $scope.message = "This is anotherController";
        });
    </script>
</body>
</html>
```

- Result:
- Message: This is myController
- Message: This is myController
- Message:
- Message: This is anotherController

AngularJS Events

- AngularJS includes certain directives which can be used to provide custom behavior on various DOM events, such as click, dblclick, mouseenter etc.
- The following table lists AngularJS event directives.

- Event Directive
- ng-blur
- ng-change
- ng-click
- ng-dblclick
- ng-focus
- ng-keydown
- ng-keyup
- ng-keypress
- ng-mousedown
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-mouseup

ng-click

- The ng-click directive is used to provide event handler for click event.
- <!DOCTYPE html>
- <html>
- <head>
- <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
- </head>

- <body ng-app="myApp" >
- <h1>AngularJS ng-click Demo: </h1>
-
- Enter Password: <input type="password" ng-model="password" />

-
- <button ng-click="DisplayMessage(password)">Show Password</button>

-
- <script> var myApp =angular.module('myApp', []);
- myApp.controller("myController", function ($scope, $window) {
- $scope.DisplayMessage = function (value)
- {
- $window.alert(value) }
- }
-);
- </script>
- </body>
- </html>

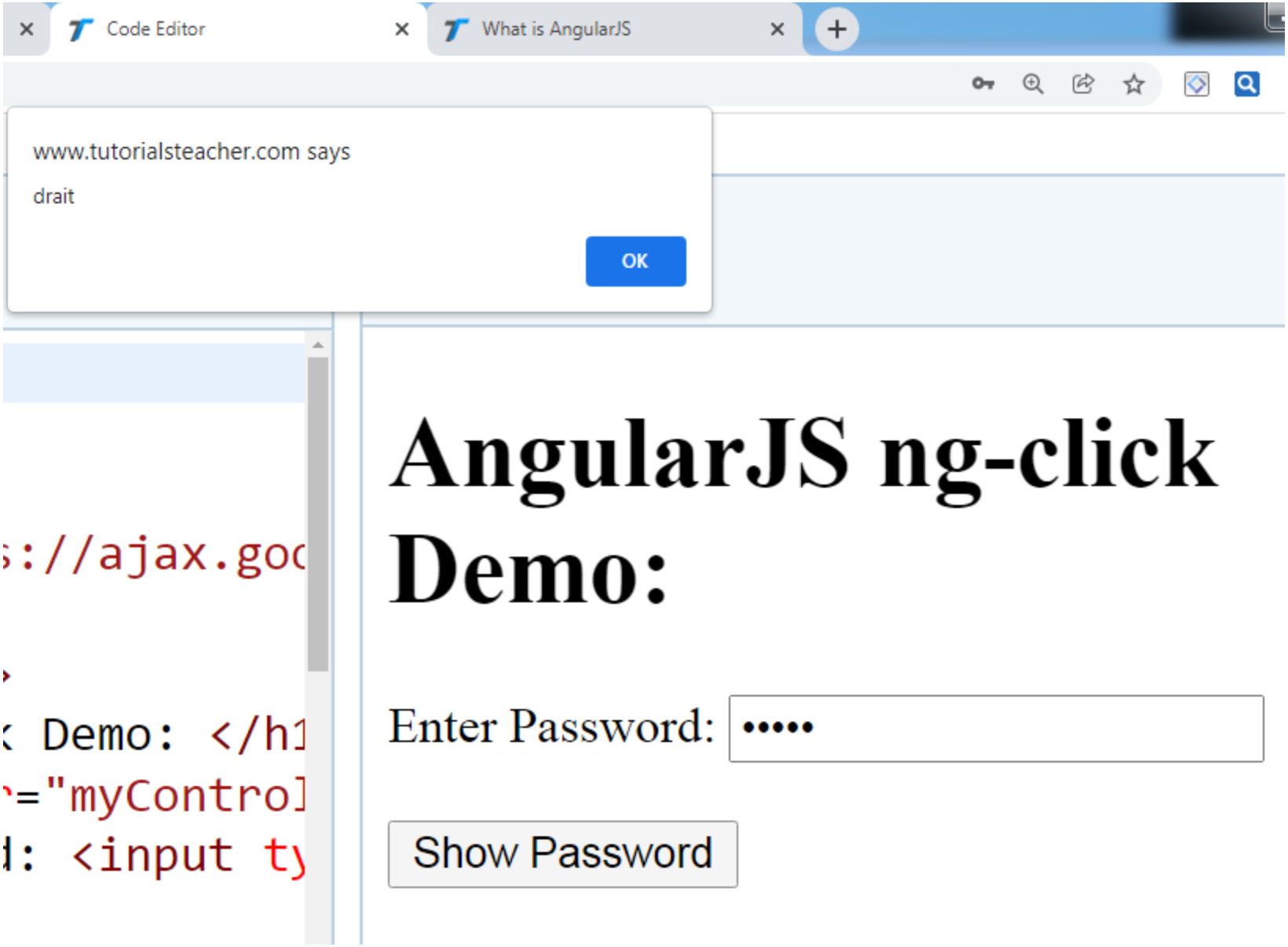
What is AngularJS

Result:

AngularJS ng-click Demo:

Enter Password:

Show Password


Code Editor
What is AngularJS
www.tutorialsteacher.com says
drait
OK
s://ajax.goo
Demo: </h1
="myControl
d: <input ty
AngularJS ng-click Demo:
Enter Password:
Show Password

AngularJS Filters

- AngularJS Filters allow us to format the data to display on UI without changing original format.
- Filters can be used with an expression or directives using pipe | sign.
- {{expression | filterName:parameter }}

Angular includes various filters to format data of different data types. The following table lists important filters.

Filter Name	Description
Number	Formats a numeric data as text with comma and fraction.
Currency	Formats numeric data into specified currency format and fraction.
Date	Formats date to string in specified format.
Uppercase	Converts string to upper case.
Lowercase	Converts string to lower case.
Filter	Filters an array based on specified criteria and returns new array.
orderBy	Sorts an array based on specified predicate expression.
Json	Converts JavaScript object into JSON string
limitTo	Returns new array containing specified number of elements from an existing array

Adding Filters to Expressions

Run

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/an
gular.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 = "John",
    $scope.lastName = "Doe"
});
</script>
```

The name is DOE

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:

Run

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

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

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

</div>

<script>
angular.module('myApp', []).controller('namesCtrl',
function($scope) {
```

Looping with objects:

- Joe, Denmark
- Birgit, Denmark
- Margareth, England
- Mary, England
- Jani, Norway
- Hege, Norway
- Kite, Norway
- Carl, Sweden
- Gustav, Sweden

Run

```
<script>
angular.module('myApp', []).controller('namesCtrl',
function($scope) {
    $scope.names = [
        {name:'Jani',country:'Norway'},
        {name:'Carl',country:'Sweden'},
        {name:'Margareth',country:'England'},
        {name:'Hege',country:'Norway'},
        {name:'Joe',country:'Denmark'},
        {name:'Gustav',country:'Sweden'},
        {name:'Birgit',country:'Denmark'},
        {name:'Mary',country:'England'},
        {name:'Kite',country:'Norway'}
        ];
});
</script>

</body>
</html>
```

Looping with objects:

- Joe, Denmark
- Birgit, Denmark
- Margareth, England
- Mary, England
- Jani, Norway
- Hege, Norway
- Kite, Norway
- Carl, Sweden
- Gustav, Sweden

The currency Filter

- The currency filter formats a number as currency:

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

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

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

</div>

<script>
var app = angular.module('myApp', []);
app.controller('costCtrl', function($scope) {
    $scope.price = 58;
});
</script>
```

```
</script>

<p>The currency filter formats a number to a currency format.
</p>

</body>
</html>
```

- Output:

Price: $58.00

The currency filter formats a number to a currency format.

AngularJS Service:

- 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:

Why use Services?

- 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.

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

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

- <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>

Today's welcome message is:

Hello AngularJS Students

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

Create Your Own Service

- To create your own service, connect your service to the module:
- Example: Create a service named hexafy:

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

- The hexadecimal value of 255 is:

- <h1>{{hex}}</h1>

-

- A custom service with a method that converts a given number into a hexadecimal number.

- <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>

output:

The hexadecimal value of 255 is:

ff

A custom service with a method that converts a given number into a hexadecimal number.

AngularJS Forms

- An AngularJS Form Example
- First Name:
-

  Last Name:
-

  RESET
- form = {"firstName":"John","lastName":"Doe"}
- master = {"firstName":"John","lastName":"Doe"}
-

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

- <div ng-app="myApp" ng-controller="formCtrl">
-   <form novalidate>
-     First Name:<br>
-     <input type="text" ng-model="user.firstName"><br>
-     Last Name:<br>
-     <input type="text" ng-model="user.lastName">
-     <br><br>
-     <button ng-click="reset()">RESET</button>
-   </form>
-   <p>form = {{user}}</p>
-   <p>master = {{master}}</p>
- </div>
```

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

</body>
</html>
```

First Name:

John

Last Name:

Doe

RESET

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

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

First Name:

John vinod

Last Name:

Doe aaaaaaaa

RESET

form = {"firstName":"John vinod","lastName":"Doe aaaaaaaa"}

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

- Example Explained
- The **ng-app** directive defines the AngularJS application.
- The **ng-controller** directive defines the application controller.
- The **ng-model** directive binds two input elements to the **user** object in the model.
- The **formCtrl** controller sets initial values to the **master** object, and defines the **reset()** method.
- The **reset()** method sets the **user** object equal to the **master** object.
- The **ng-click** directive invokes the **reset()** method, only if the button is clicked.

AngularJS Form Validation

- AngularJS can validate input data
- AngularJS offers client-side form validation.
- E-mail validation:

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

- Try writing an E-mail address in the input field:

- <form name="myForm">
- <input type="email" name="myInput" ng-model="myInput">
- </form>

- The input's valid state is:
- <h1>{{myForm.myInput.$valid}}</h1>
- Note that the state of the input field is "true" before you start writing in it, even if it does not contain an e-mail address.

- </body>
- </html>

Try writing an E-mail address in the input field:

The input's valid state is:

true

Note that the state of the input field is "true" before you start writing in it, even if it does not contain an e-mail address.

Try writing an E-mail address in the input field:

vinodkumar940

The input's valid state is:

false

Note that the state of the input field is "true" before you start writing in it, even if it does not contain an e-mail address.

9:44 AM
12/13/2021

Try writing an E-mail address in the input field:

vinodkumar940@gmail.com

The input's valid state is:

true

Note that the state of the input field is "true" before you start writing in it, even if it does not contain an e-mail address.