HTML 5, CSS 3 with Bootstrap, Javascript ES6. Angular JS: Introduction to HTML 5, CSS 3 with Bootstrap, Javascript ES6 Getting started with Node.js, Introduction To Angular JS, Angular JS Modules, Directives & Building Databases.

**Introduction To Angular JS, Angular JS Modules, Directives & Building Databases.**

**Introduction To Angular JS:**

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 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>

## Steps to Create First Angular App

**Example:**This example illustrates the basic Hello World app using Angular JS.

// Filename - app.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'my-app',

template: '<h1>Hello World!</h1>',

styleUrls: ['./app.component.css']})

export class AppComponent {}

# AngularJS Modules

The **AngularJS module** defines the functionality of the application which is applied on the entire HTML page. It helps to link many components. So it is just a group of related components. It is a container that consists of different parts like controllers, services, and directives.

**Note:**These modules should be made in normal HTML files like index.html and no need to create a new project in VisualStudio for this section.

**Creating a Module in AngularJS:**

var app = angular.module("Module-name", []);

In this [], we can add a list of components needed but we are not including any components in this case. This created module is bound with any tag like div, body, etc by adding it to the list of modules.

<div ng-app = "module-name">

The code in which the module is required.

</div>

**Adding a Controller:**

app.controller("Controller-name", function($scope) {

$scope.variable-name= "";

});

<!DOCTYPE html>

<html ng-app="myApp">

<head>

    <title>AngularJS Module Example</title>

    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

    <script>

        // Define a module named 'myApp'

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

        // Define a controller 'MyController' attached to 'myApp' module

        app.controller('MyController', function($scope) {

            $scope.message = 'Hello, AngularJS!';

        });

    </script>

</head>

<body>

    <!-- Use the 'myApp' module and 'MyController' controller -->

    <div ng-controller="MyController">

        <h1>{{ message }}</h1>

    </div>

</body>

</html>

### Directives in AngularJS

Directives in AngularJS are markers on a DOM element that tell AngularJS's HTML compiler ($compile) to attach a specified behavior to that DOM element or even transform the DOM element and its children. Directives are a powerful feature in AngularJS for extending HTML with new attributes and elements that can encapsulate complex behaviors.

#### Example of Using a Directive

Here's a simple example of creating a custom directive named helloDirective that displays a greeting message:

<!DOCTYPE html>

<html ng-app="myApp">

<head>

    <title>AngularJS Directives Example</title>

    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

    <script>

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

        // Define a custom directive named 'helloDirective'

        app.directive('helloDirective', function() {

            return {

                template: '<h3>Hello, Custom Directive!</h3>'

            };

        });

    </script>

</head>

<body>

    <div ng-controller="MyController">

        <!-- Use the 'helloDirective' directive -->

        <hello-directive></hello-directive>

    </div>

    <script>

        // Define a controller 'MyController'

        app.controller('MyController', function($scope) {

            // Controller logic can go here if needed

        });

    </script>

</body>

</html>

# **AngularJS Directives**

AngularJS facilitates you to extend HTML with new attributes. These attributes are called directives.

There is a set of built-in directives in AngularJS which offers functionality to your applications. You can also define your own directives.

Directives are special attributes starting with ng- prefix. Following are the most common directives:

* ng-app: This directive starts an AngularJS Application.
* ng-init: This directive initializes application data.
* ng-model: This directive defines the model that is variable to be used in AngularJS.
* ng-repeat: This directive repeats html elements for each item in a collection.

## ng-app directive

ng-app directive defines the root element. It starts an AngularJS Application and automatically initializes or bootstraps the application when web page containing AngularJS Application is loaded. It is also used to load various AngularJS modules in AngularJS Application.

<div ng-app = "">

   ...

</div>

## ng-init directive

ng-init directive initializes an AngularJS Application data. It defines the initial values for an AngularJS application.

In following example, we'll initialize an array of countries. We're using JSON syntax to define array of countries.

<div ng-app = "" ng-init = "countries = [{locale:'en-IND',name:'India'}, {locale:'en-PAK',name:'Pakistan'}, {locale:'en-AUS',name:'Australia'}]">

   ...

</div>

## ng-model directive:

ng-model directive defines the model/variable to be used in AngularJS Application.

In following example, we've defined a model named "name".

<div ng-app = "">

   ...

   <p>Enter your Name: <input type = "text" ng-model = "name"></p>

</div>

## ng-repeat directive

ng-repeat directive repeats html elements for each item in a collection. In following example, we've iterated over array of countries.

<div ng-app = "">

   ...

   <p>List of Countries with locale:</p>

   <ol>

      <li ng-repeat = "country in countries">

         {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}

      </li>

   </ol>

## AngularJS directives Example

<!DOCTYPE html>

<html>

<head>

      <title>AngularJS Directives</title>

</head>

<body>

      <h1>Sample Application</h1>

      <div ng-app = "" ng-init = "countries = [{locale:'en-IND',name:'India'}, {locale:'en-PAK',name:'Pakistan'}, {locale:'en-AUS',name:'Australia'}]">

         <p>Enter your Name: <input type = "text" ng-model = "name"></p>

         <p>Hello <span ng-bind = "name"></span>!</p>

         <p>List of Countries with locale:</p>

         <ol>

            <li ng-repeat = "country in countries">

               {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}

            </li>

         </ol>

      </div>

<script src = "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

</body>

</html>

Html:

!DOCTYPE html>

<html>

<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

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

<script>

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

app.directive("w3TestDirective", function() {

    return {

        template : "This is a directive constructor. "

    };

});

</script>

</body>

</html>

### Building Databases in AngularJS Context

AngularJS itself does not directly interact with databases. Typically, AngularJS is used on the frontend to create interactive user interfaces. For database interactions, you would typically use AngularJS in conjunction with a backend server (e.g., Node.js, Python Flask, Java Spring Boot) that handles database operations.

#### Example Scenario with Node.js and MongoDB

Here's a high-level example of how AngularJS can interact with a backend (Node.js with Express) that handles database operations (MongoDB):

// Example using Express.js and MongoDB

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.json());

mongoose.connect('mongodb://localhost/mydatabase', { useNewUrlParser: true, useUnifiedTopology: true });

const CarSchema = new mongoose.Schema({

    make: String,

    model: String,

    year: Number

});

const Car = mongoose.model('Car', CarSchema);

// Example API endpoints

app.post('/api/cars', (req, res) => {

    const newCar = new Car(req.body);

    newCar.save((err) => {

        if (err) {

            res.status(500).send('Error saving car to database');

        } else {

            res.status(200).send('Car saved successfully');

        }

    });

});

app.get('/api/cars', (req, res) => {

    Car.find({}, (err, cars) => {

        if (err) {

            res.status(500).send('Error fetching cars from database');

        } else {

            res.status(200).send(cars);

        }

    });

});

app.listen(3000, () => {

    console.log('Server running on http://localhost:3000');

});

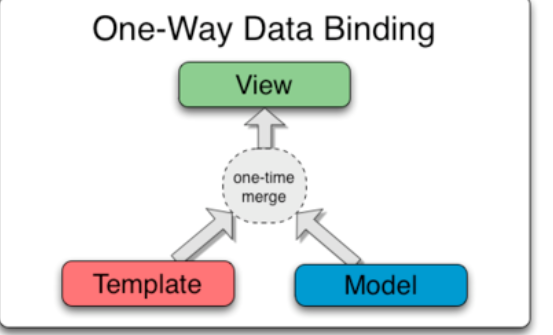
# **AngularJS Data Binding**

Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application.

AngularJS follows Two-Way data binding model.

## One-Way Data Binding

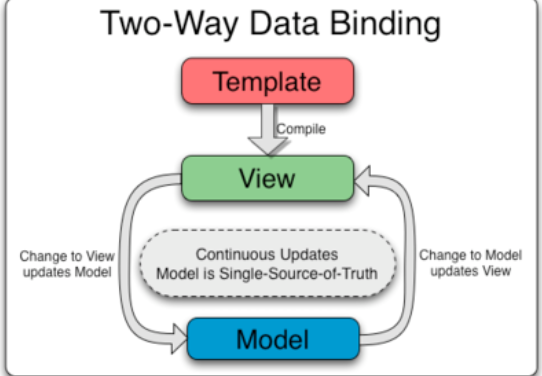
The one-way data binding is an approach where a value is taken from the data model and inserted into an HTML element. There is no way to update model from view. It is used in classical template systems. These systems bind data in only one direction.



## Two-Way Data Binding

Data-binding in Angular apps is the automatic synchronization of data between the model and view components.

Data binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. If the model is changed, the view reflects the change and vice versa.



<!DOCTYPE html>

<html>

<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

<div ng-app="" ng-init="firstName='Ajeet'">

<p>Input something in the input box:</p>

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

<p>You wrote: {{ firstName }}</p>

</div>

</body>

</html>

In the above example, the {{ firstName }} expression is an AngularJS data binding expression. Data binding in AngularJS binds AngularJS expressions with AngularJS data.

{{ firstName }} is bound with ng-model="firstName".

Let's take another example where two text fields are bound together with two ng-model directives:

<!DOCTYPE html>

<html>

<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

<div data-ng-app="" data-ng-init="quantity=1;price=20">

<h2>Cost Calculator</h2>

Quantity: <input type="number" ng-model="quantity">

Price: <input type="number" ng-model="price">

<p><b>Total in rupees:</b> {{quantity \* price}}</p>

</div>

</body>

</html>