**Assignment-3**

1. **What is Angular JS with ng directives?**

* **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.

|  |
| --- |
| **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 ng-bind="name"></p> </div> </body> </html> |

1. **Write & Explain Setup Angular JS 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](https://angularjs.org/) -> click download button, which will open the following popup.
* **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](http://plnkr.co/), [jsbin.com](http://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 useGoogle Chrome while developing an application.

1. **Explain with diagram MVC Architecture?**

|  |
| --- |
|  |

* **MVC Architecture:** Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. MVC is popular because it isolates the application logic from the user interface layer and supports separation of concerns. The controller receives all requests for the application and then works with the model to prepare any data needed by the view. The view then uses the data prepared by the controller to generate a final presentable response.
* **The MVC pattern is made up of the following three parts:**
* **Model:**It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.
* **View:**It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
* **Controller:**It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

1. **Short note on Directive & Why it uses in Angular-JS?**

* **Angular-JS Directive:** Directives are markers on the DOM element which tell AngularJS to attach a specified behavior to that DOM element or even transform the DOM element with its children. Simple Angular-JS allows extending HTML with new attributes called Directives. Angular-JS has a set of built-in directives which offers functionality to the applications. It also defines its own directives. A directive can be defined using some functions which are: Element name, Attribute, Class, and Comment.
* **Why use Directive in Angular-JS:**
* It gives support to creating custom directives for different types of elements.
* A directive is activated when the same element or matching element is there in front.
* It is used to give more power to HTML by helping them with the new syntax.
* Directive classes, like component classes, can implement life-cycle hooks to influence their configuration and behavior.

1. **Which are Directive Components in Angular-JS?**

* **The Angular-JS directives extend the attribute with the prefix ng-. Some directive components are discussed below:**
  1. [**Ng-app**](https://www.geeksforgeeks.org/angularjs-ng-app-directive/)**:**The ng-app directive is used to define the root element of an Angular-JS application. This directive automatically initializes the Angular-JS application on page load. **Example:**This example illustrates the implementation of the ng-app directive in Angular-JS.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head> <title> AngularJS ng-app Directive </title>  <script src="<https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js>"> </script> </head>  <body style="text-align:center">      <h1 style="color:green">More Group’s</h1>      <h3 style="color:green">ng-app directive</h3>      <div ng-app="" ng-init="name=' More Group’s '">   <p>{{ name }} is the portal for More Group’s.</p> </div>  </body> </html> |

* 1. [**Ng-controller**](https://www.geeksforgeeks.org/angularjs-ng-controller-directive/)**:** The ng-controller Directive in AngularJS is used to add the controller to the application. It can be used to add methods, functions, and variables that can be called on some event like a click, etc to perform a certain action. **Example:** This example illustrates the implementation of the ng-controllerdirective in AngularJS.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head> <title>AngularJS ng-controller Example</title>  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>  </head>  <body ng-app="myApp">  <div ng-controller="MyController">  <h1>{{ greeting }}</h1> <button ng-click="changeGreeting()">Change Greeting</button>  </div>  <script>  var app = angular.module('myApp', []);  app.controller('MyController', function($scope) {  $scope.greeting = 'Hello, World!';  $scope.changeGreeting = function() {  $scope.greeting = 'New Greeting!';  }; });  </script>  </body></html> |

* 1. [**Ng-bind**](https://www.geeksforgeeks.org/angularjs-ng-bind-directive/)**:**The ng-bind directiveis used to bind/replace the text content of a particular element with the value that is entered in the given expression. The value of specified HTML content updates whenever the value of the expression changes in the ng-bind directive. **Example:** This example illustrates the implementation of the ng-binddirective in AngularJS.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head> <title>AngularJS ng-bind Example</title>  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>  </head>  <body ng-app="myApp">  <div ng-controller="MyController">  <h1 ng-bind="greeting"></h1> <button ng-click="changeGreeting()">Change Greeting</button> </div>  <script>  var app = angular.module('myApp', []);  app.controller('MyController', function($scope) {  $scope.greeting = 'Hello, World!';  $scope.changeGreeting = function() {  $scope.greeting = 'New Greeting!';  };  });  </script>  </body>  </html> |

1. **Describe AngularJS Expressions with example?**

* **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 }}

|  |
| --- |
| <!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> |

1. **Explain AngularJS Controllers with example?**

* **AngularJS controllers** are used to control the flow of data of AngularJS application. A controller is defined using ng-controller directive. A controller is a JavaScript object containing attributes/properties and functions. Each controller accepts $scope as a parameter which refers to the application/module that controller is to control. The ng-controller directive defines the application controller. A controller is a JavaScript Object, created by a standard JavaScript object constructor.
* **Use controllers to:** Set up the initial state of the $scope object, Add behavior to the $scope object.
* **Do not use controllers to:**
* **Manipulate DOM:** Controllers should contain only business logic. Putting any presentation logic into Controllers significantly affects its testability. AngularJS has databinding for most cases and directives to encapsulate manual DOM manipulation.
* **Format input:** Use AngularJS form controls instead.
* **Filter output:** Use AngularJS filters instead.
* **Share code or state across controllers:** Use AngularJS services instead. Manage the life-cycle of other components (for example, to create service instances).
* **Angular-JS Controller Example:** The example above demonstrated a controller object with two properties: lastName and firstName. A controller can also have methods (variables as functions):

|  |
| --- |
| <!DOCTYPE html>  <html>  <head> <title>AngularJS User Information Example</title>  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>  </head>  <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 = "Aryan";  $scope.lastName = "Khanna"; });  </script>  </body></html> |

1. **Life Cycle of Angular-JS Scope with diagram?**

* **Life Cycle of Angular-JS Scope:** In the flow of javascript code, whenever an event is received by browser then it executes a javascript callback corresponding to it. --And after the completion of callback, DOM objects are re-rendered by a browser. If any javascript code is executed outside the context of angularJS by browser then angularJS remain unaware with the changes in the model. To detect the model change in angularJS $apply API is used.
* **Importance of Scope lifecycle:** This lifecycle is used right from the creation to the destruction phase and thus makes it easier for the developer to create applications. Since it uses functions like $apply(), and $digest() it protects the application from any security mishap. Model mutation acts as a boon for developers in Angular as it monitors the code as well as updates and reflects the changes made. Whenever any child scope is not required, it is destroyed which lowers the load on the server as well as the chunk of memory allocated to the scope is made free.

|  |
| --- |
| **Life cycle of Scope in AngularJS** |

* **Life Cycle of AngularJS Scope:**

1. **Creation:** During bootstrap of an application using $injector, root scope is created. And during the linking of a template, many of the directives creates new child scope.
2. **Watcher Registration:** Model values can propagate to DOM using a $watch
3. **Model Mutation:** To observe mutation in a proper way, an API known as $apply is used.
4. **Mutation Observation:** After $apply, a $digest is performed on the root scope in angular JS.
5. **Scope Destruction:**  If child scope is no longer in use, then it should be destroyed. Child scope creator is responsible to destroy the child scope if no longer in use. Using API $scope.$destroy() child scope creator can do so. Destroying an unused child scope will release memory that is being used by it and stop model propagation too.
6. **Explain in detail AngularJS Data Binding with example?**

* **AngularJS Data Binding:** Data binding in AngularJS is the synchronization between the model and the view. AngularJS implements data-binding that treats the model as the single-source-of-truth in your application & for all the time, the view is a projection of the model.

1. **One-way Binding:**This type of binding is unidirectional, i.e. this binds the data flow from either component to view(DOM) or from the view(DOM) to the component. There are various techniques through which the data flow can be bind from component to view or vice-versa. If the data flow from component to view(DOM), then this task can be accomplished with the help of String Interpolation & Property Binding.

* [**Interpolation**](https://www.geeksforgeeks.org/string-interpolation-in-angular-8/)**:**Angular interpolation is used to display a component property in the respective view template with double curly braces syntax. Interpolation is used to transfer properties mentioned in the component class to be reflected in its template.
* [**Property Binding**](https://www.geeksforgeeks.org/property-binding-in-angular-8/)**:** Similar to Java, variables defined in the parent class can be inherited by the child class which is templates in this case. So if we have to store Boolean or other data types then use Property Binding.

|  |
| --- |
| **Syntax:** class="{{variable\_name}}" |
| **Example:**  <h3>Binding Types</h3>    <p>Interpolation</p> <br>  <h5>        Addition of 3 and 5 with        Interpolation is {{3+5}}  </h5>  <h5>        Addition of 3 and 5 without        Interpolation is 3+5  </h5>  <h2>{{val}}</h2> |

|  |
| --- |
| https://docs.angularjs.org/img/One_Way_Data_Binding.png |

1. [**Event Binding**](https://www.geeksforgeeks.org/event-binding-in-angular-8/)**:** An event is created whenever either a key is pressed or on a mouse clicked. It is used to handle the events raised by the user actions like button click, mouse movement, keystrokes, etc. When the DOM event happens at an element(e.g. click, keydown, keyup), it calls the specified method in the particular component.Using Event Binding, we can bind data from DOM to the component and hence can use that data for further purposes.

|  |
| --- |
| <h3>Binding Types</h3>  <p>Event Binding</p>  <button class="btn btn-block" (click)="Clickme($event)"> Click Here </button> |

1. [**Two way Binding**](https://www.geeksforgeeks.org/event-binding-in-angular-8/)**:** In this type of binding, the immediate changes to the view & component, will be reflected automatically, i.e. when the changes made to the component or model then the view will render the changes simultaneously. Similarly, when the data is altered or modified in the view then the model or component will be updated accordingly.

|  |  |
| --- | --- |
| div style="text-align: center">        <h1 style="color: green">          GeeksforGeeks        </h1>        <h3>Two-way Data Binding</h3>        <input type="text"          placeholder="Enter text"          [(ngModel)]="val" />        <br />        {{ val }}  </div> | **https://docs.angularjs.org/img/Two_Way_Data_Binding.png** |

1. **Describe AngularJS component?**

* **A component** is a directive that uses a more straightforward configuration that fits a component-based architecture, which Angular 2 is all about. Think of an element as a widget: A piece of HTML code that you can reuse in various places in your web application.
* **Components:** app.component.css, app.component.html, app.component.spec.ts, app.component.ts, app.module.ts. These are the component files that get created by default when you create an application in Angular.
* **Types of Components in Angular:** Parent, Child. The Parent Component is the one that is predefined and imported when you create the App. However, the Child component is the one that you can design according to your needs and demand.
* **Metadata of Angular Component:**
* **There is three-component configuration**

|  |
| --- |
| import {Component} from ‘@angular/core’; @Component({   Selector: ‘app-root’,   templateUrl: ‘./app.component.html’,   providers: [”] }) export class AppComponent {   title=‘ExternLabs’; } |

* **Angular components overview:** Components are the main building blocks for Angular applications.Each component consists of:
* An HTML template that declares what renders on the page
* A TypeScript class that defines behavior
* A CSS selector that defines how the component is used in a template Optionally, CSS styles applied to the template.
* In AngularJS, a Component is a special kind of directive that uses a simpler configuration which is suitable for a component-based application structure. This makes it easier to write an app in a way that's similar to using Web Components or using the new Angular's style of application architecture.

1. **Short note on Angular js services?**

* **The**[**Services**](https://docs.angularjs.org/guide/services)is a function or an object that avails or limit to the application in AngularJS ie., it is used to create variables/data that can be shared and can be used outside the component in which it is defined. Service facilitates built-in service or can make our own service. The Service can only be used inside the controller if it is defined as a dependency.
* **Why to use the AngularJS Service:** AngularJS supervise the application constantly. In order to handle the events or any changes in a proper manner, then the Service that is provided by the AngularJS will prefer to use, instead of Javascript Objects.
* **There are some commonly used built-in services, are described below:**
* **$http Service:** It makes the request to the server, in order to handle the response by the application.
* **$timeout Service:** This service is AngularJS’ version of the window.setTimeout function.
* **$interval Service:**This service is AngularJS’ version of the window.setInterval function.
* **Create the AngularJS Service:**
* **STEP 1:**Creating a service will follow the below command: **ng g s service-name.** s is a short form for service. This creates two files service-name.service.spec.ts which is not supposed to be changed and service-name.service.ts.
* **STEP 2:**After the service is created, we have to include it in the providers of app.module.ts, providers: **[Service-nameService],** Here, the first letter of the service-name should be capitalized followed by Service written without any space.
* **STEP 3:**So we have to now make changes in service-name.service.ts to create a JSON variable that is supposed to be made available to various components, **Sailors = [22, ‘Dustin’, 7];** The sailors variable here is an array.
* **STEP 4:**In app.component.ts make the following changes:
* import the service among the rest of the required imports. **Example: import { Service-nameService } from './service-name.service';** just like the way we did it in providers.
* **STEP 5:**In app.component.html we will print the data stored in newData: **{{newData}}**
* **Note:** As we have added ngFor in app.component.html we will have to import FormsModule in app.module.ts

1. **Explain in brief AngularJS Dependency Injection with example?**

* **AngularJS Dependency Injection:** AngularJS comes with a built-in dependency injection mechanism. It facilitates you to divide your application into multiple different types of components which can be injected into each other as dependencies. Dependency Injection is a software design pattern that specifies how components get holds of their dependencies. In this pattern, components are given their dependencies instead of coding them within the component. Modularizing your application makes it easier to reuse, configure and test the components in your application. It provides following core components which can be injected into each other as dependencies: Value, Factory, Service, Provider, Constant.
* **Value:** Value is a simple JavaScript object, which is required to pass values to the controller during config phase (config phase is when AngularJS bootstraps itself).

|  |
| --- |
| //define a module  var mainApp = angular.module("mainApp", []);  //create a value object as "defaultInput" and pass it a data.  mainApp.value("defaultInput", 5);  //inject the value in the controller using its name "defaultInput"  mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {  $scope.number = defaultInput;  $scope.result = CalcService.square($scope.number);  $scope.square = function() {  $scope.result = CalcService.square($scope.number);  } }); |

* **Factory:** Factory is a function which is used to return value. It creates a value on demand whenever a service or a controller requires it. It generally uses a factory function to calculate and return the value..

|  |
| --- |
| //define a module  var mainApp = angular.module("mainApp", []);   //create a factory "MathService" which provides a method multiply to return multiplication of two numbers  mainApp.factory('MathService', function() {  var factory = {};  factory.multiply = function(a, b) {  return a \* b  }  return factory;  });   //inject the factory "MathService" in a service to utilize the multiply method of factory.  mainApp.service('CalcService', function(MathService) {  this.square = function(a) {  return MathService.multiply(a,a);  } }); |

* **Service:** Service is a singleton JavaScript object containing a set of functions to perform certain tasks. Service is defined using service() function and it is then injected into the controllers.

|  |
| --- |
| //define a module  var mainApp = angular.module("mainApp", []);  //create a service which defines a method square to return square of a number.  mainApp.service('CalcService', function(MathService) {  this.square = function(a) {  return MathService.multiply(a,a); }  });   //inject the service "CalcService" into the controller  mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {  $scope.number = defaultInput;  $scope.result = CalcService.square($scope.number);  $scope.square = function() {  $scope.result = CalcService.square($scope.number);  } }); |

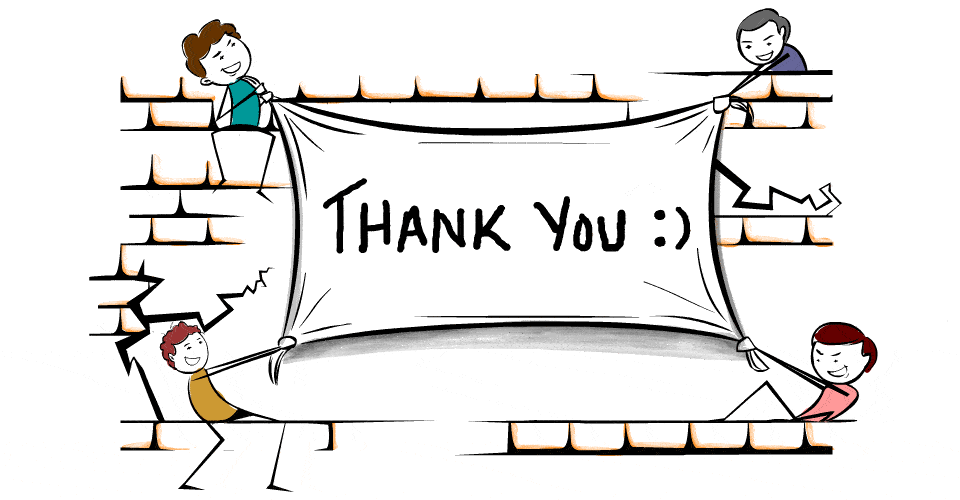
* **Provider:** Provider is used by AngularJS internally to create services, factory, etc. during the config phase. The following script can be used to create MathService that we created earlier. --Provider is a special factory method with get() method which is used to return the value/service/factory.

|  |
| --- |
| //define a module  var mainApp = angular.module("mainApp", []);   //create a service using provider which defines a method square to return square of a number.  mainApp.config(function($provide) {  $provide.provider('MathService', function() {  this.$get = function() {  var factory = {};  factory.multiply = function(a, b) {  return a \* b;  }  return factory;  };  });  }); |

* **Constant:** Constants are used to pass values at the config phase considering the fact that value cannot be used during the config phase.

|  |
| --- |
| mainApp.constant("configParam", "constant value") |

**(PPT Give you more understanding than PDF)** The material for the PDF has been compiled from various sources such as books, tutorials (offline and online), lecture notes, several resources available on Internet. The information contained in this PDF is for general information and education purpose only. While we endeavor to keep the information up to date and correct, **we make no representation of any kind about the completeness and accuracy of the material.** The information shared through this PDF material should be used for educational purpose only.

****