Angular JS

What is AngularJS?

- 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**.
- Developed by google and released 2010

What is AngularJS?

- 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>
```

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

Why AngularJS?

- Two-way data binding
- MVC architecture
- Dependency Injection
- Directives extend HTML

MVC Pattern in AngularJS

- Model → Data
  View → HTML
  Controller → Logic
- Draw diagram:
- User → View → Controller → Model → View updates

How AngularJS Works

- Browser loads HTML
- AngularJS bootstraps app
- Creates scope
- Binds data to DOM
- Watches for changes
- Mention briefly:
  Service Portal uses this same mechanism internally.

AngularJS Applications

- AngularJS **modules** define AngularJS applications.
- AngularJS **controllers** control AngularJS applications.
- The **ng-app** directive defines the application, the **ng-controller** directive defines the controller.

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

<p>Try to change the names.</p>

<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>
```

Creating a Module

- Module is the container of application components.
- var app = angular.module("myApp", []);
- Key Points:

First parameter = name

Second parameter = dependencies

- **ng-app Directive :** Defines AngularJS application root.

Explain:
This tells Angular where the app starts.

Note: Only one ng-app is usually used.

Controllers

Controller Responsibilities:

- Handle UI logic
- Manage data
- Connect model and view

Explain:
Controller is like brain of the UI.

```
app.controller("mainCtrl", function($scope){
    $scope.name = "Sandy";
});
```

- **ng-controller**

```
<div ng-controller="mainCtrl">
  {{ name }}
</div>
```

Explain:
Controller attaches data to scope.

Controllers

What is $scope?

- Communication bridge between controller and view
- Stores variables and functions

Explain visually:
Controller → $scope → View

Data Binding

What is Data Binding?

- Automatic synchronization between model and view.
- Explain:
- If model changes → UI updates
  If UI changes → model updates

This is **core AngularJS concept**.

Expressions {{ }}

- Used to display data.

```
{{ 5 + 5 }}
{{ name }}
```

Data Binding

- **Two-Way Data Binding**

<input ng-model="name">
{{ name }}

Explain:

Change input → view updates

Change model → input updates

VERY IMPORTANT concept.

- **One-Way vs Two-Way Binding**

Explain difference clearly.

Traditional JS:
Manual DOM update.

AngularJS:
Automatic update.

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

<div data-ng-app="" data-ng-init="quantity=1;price=5">

<h2>Cost Calculator</h2>

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

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

</div>

</body>
</html>
```

Directives

- **Built-in Directives**

What is a Directive?

- Directive is an HTML attribute that extends HTML functionality.
- Starts with ng-
- Extends HTML behavior

Directives

ng-model

- 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.

ng-click

- used for Event handling
- <button ng-click="count = count + 1">Click</button>

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-focus
- ng-keydown
- ng-keypress
- ng-keyup
- ng-mousedown
- ng-mouseenter
- ng-mouseover
- ng-mouseup

```
<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>
```

Directives

Example for functions create

```
<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>
```

Directives

another 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="mainCtrl">
<input type="text" ng-model="name" placeholder="Enter Name">
<br><br>
<input type="email" ng-model="email" placeholder="Enter Email">
<br><br>
<button ng-click="showData()">
Show Data
</button>
<hr>
<h3>Output</h3>
<p>Name: {{ outputName }}</p>
<p>Email: {{ outputEmail }}</p>

</div>
<script>
    var app = angular.module("myApp", []);
app.controller("mainCtrl", function($scope){
    $scope.name = "";
    $scope.email = "";
    $scope.showData = function(){
        $scope.outputName = $scope.name;
        $scope.outputEmail = $scope.email;
    }
});
</script>
</body>
</html>
```

Directives

The ng-disabled Directive

- The **ng-disabled** directive binds AngularJS application data to the disabled attribute of HTML elements

```
<div ng-app="" ng-init="mySwitch=true">
    <p>
        <button ng-disabled="mySwitch">Click Me!</button>
    </p>
    <p>
        <input type="checkbox" ng-model="mySwitch">Button
    </p>
    <p>{{ mySwitch }}</p>
</div>
```

Directives

- **ng-show / ng-hide**

Conditional visibility.

```
<form ng-app="" name="myForm">
  Email:
  <input type="email" name="myAddress" ng-model="text">
  <span ng-show="myForm.myAddress.$error.email">Not a valid e-mail address</span>
</form>
```

- **ng-if**

Difference between:

ng-if (removes from DOM)

ng-show (just hides)

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

<button ng-click="showMessage()">
Show Message
</button>

<h3 ng-if="isShown">
Hello Students
</h3>

</div>
<script>
var app = angular.module("myApp", []);
app.controller("mainCtrl", function($scope){

    $scope.isShown = false;
    $scope.showMessage = function(){
        $scope.isShown = true;
    };

});
</script>
```

Directives

- **ng-repeat :**Very important slide.
- Used for: Looping arrays.
- Explain:
  Looping over arrays.

```
<div ng-app="myApp" ng-controller="mainCtrl">
<h3>Student List</h3>
<ul>
    <li ng-repeat="student in students">
        {{ student }}
    </li>
</ul>
</div>
<script>
var app = angular.module("myApp", []);
app.controller("mainCtrl", function($scope){
    $scope.students = ["Ali","Sara","Omar","Mona"];
});
</script>
```

Task

- Build a simple Login Form UI using **AngularJS directives** with good design.
- **Requirements:**
- Create a login form that contains:
  - Student Name input
  - Password input
  - Checkbox
- Add a Login Button:
  - When clicked, copy the student name to an output variable using ng-click.
- Implement form validation:
  - Both username and password fields are required.
  - Disable the login button if the form is invalid.
- Display Section:
  - Show the student name using ng-if or ng-show/ng-hide.
  - Display a message only if the checkbox is selected.
- Use simple and good clean UI design using bootstap.

Arrays

- **What is an Array?**
- An array is a collection of items.
- students = ["Ali", "Sara", "Omar"];
- **Define Array in Controller**

```
app.controller("mainCtrl", function($scope){
    $scope.students = ["Ali", "Sara", "Omar"];
});
```

- **Display Using ng-repeat**

```
<li ng-repeat="student in students">
  {{ student }}
</li>
```

Arrays

- **Adding Item to Array**

```
$scope.newStudent = "";

$scope.addStudent = function(){
    $scope.students.push($scope.newStudent);
    $scope.newStudent = "";
};


<input type="text" ng-model="newStudent">
<button ng-click="addStudent()">Add</button>
```

Arrays

- **Removing Item from Array**
- ```
  $scope.removeStudent = function(index){
      $scope.students.splice(index, 1);
  };
  ```
- ```
  <li ng-repeat="student in students track by $index">
    {{ student }}
    <button ng-click="removeStudent($index)">Delete</button>
  </li>
  ```

Objects

- **What is an Object?**
- An object stores data in key-value pairs.
- student = {
  name: "Ali",
  age: 20,
  grade: "A"
  };
- **Define Object**

```
$scope.student = {
    name: "Ali",
    age: 20,
    grade: "A"
};
```

- **Display Object Data**

```
<p>Name: {{ student.name }}</p>
<p>Age: {{ student.age }}</p>
```

Objects

- **Array of Objects**

```
$scope.students = [
    { name: "Ali", age: 20 },
    { name: "Sara", age: 22 },
    { name: "Omar", age: 21 }
];
```

```
<tr ng-repeat="s in students">
  <td>{{ s.name }}</td>
  <td>{{ s.age }}</td>
</tr>
```

Forms

- **Forms and Validation (AngularJS)**

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

- **Simple Form Example**

```
<form name="loginForm" novalidate>
  <input type="text"
         name="username"
         ng-model="user.name"
         required>
  <input type="password"
         name="password"
         ng-model="user.password"
         required>
  <button ng-click="login()"
          ng-disabled="loginForm.$invalid">
    Login
  </button>
</form>
```

- **Show Validation Messages**

```
<div ng-show="loginForm.username.$touched &&
 loginForm.username.$invalid">
  Username is required
</div>
```

Forms

Form Validation

- AngularJS offers client-side form validation.
- AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.
- AngularJS also holds information about whether they have been touched, or modified, or not.
- You can use standard HTML5 attributes to validate input, or you can make your own validation functions.

Important form properties of validation:

Property	Meaning
`$valid`	Form is valid
`$invalid`	Form is invalid
`$touched`	User clicked input
`$dirty`	User changed input

Forms

Example : Email Validation

```
<input type="email" name="email"  ng-model="user.email"  required>

<div ng-show="loginForm.email.$error.email">
  Invalid email format
</div>
```

Forms

$event Object calling in function

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

```
<div ng-app="myApp" ng-controller="myCtrl">
    <form ng-submit="submitForm($event)">
        <input type="text" ng-model="username" placeholder="Username" required />
        <button type="submit">Login</button>
    </form>
    <p>{{message}}</p>
</div>
<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope){
    $scope.submitForm = function(event){
        event.preventDefault();
        if(!$scope.username){
            $scope.message = "Username is required";
            return;
        }
        $scope.message = "Login submitted: " + $scope.username;
    };
});
</script>
```

Task

- Create a simple AngularJS application called **Student List App**.
- Your application must allow users to add and manage student names dynamically

Requirements

- Create an AngularJS application
- Add a text input field where the user can enter a student name and set validations
- Add a button labeled **“Add Student”**. When clicked:
  - The entered name should be added to a list.
  - The input field should be cleared after adding.
  - The button must be disabled if the input is empty or invalid.
- Display the list of students using ng-repeat.
  - Each student name should appear clearly in the list.
  - Each student must have a **Delete** button next to it.
  - When the Delete button is clicked:
  - The selected student must be removed from the list.
- Show a message if no students are added:
  - Example: “No students added yet.”