Name - Gaurang A Raorane Div - D15A Roll no - 49 Batch - C

**Experiment - 3**

**Aim -** To study AngularJS

Problem Statement-

* 1. Demonstrate with an AngularJS code one way data binding and two way data binding in AngularJS
  2. Implement a basic authentication system for a web application using AngularJS. Create a simple login page that takes a username and password, and upon submission, checks for a hardcoded set of credentials. If the credentials are valid, display a success message; otherwise, show an error message.Demonstrate AngularJS controller, module and form directives.
  3. Services
  4. filters

**Theory:-**

**AngularJS** can be understood as a powerful front-end framework that simplifies and enhances web development. AngularJS extends the capabilities of JavaScript by providing a comprehensive set of tools and features for building dynamic, single-page web applications. It introduces concepts like two-way data binding, which automatically synchronizes data between the model and the view, eliminating the need for manual DOM manipulation. Additionally, AngularJS offers built-in directives for creating reusable components, form validation, and routing, streamlining the development process. Overall, AngularJS empowers developers to create interactive, responsive web applications with less code and complexity compared to traditional JavaScript development.

1. What are directives? Name some of the most commonly used directives in AngularJS application

At a high level, directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's HTML compiler ([$compile](https://docs.angularjs.org/api/ng/service/$compile)) to attach a specified behavior to that DOM element (e.g. via event listeners), or even to transform the DOM element and its children.

AngularJS comes with a set of these directives built-in, like ngBind, ngModel, and ngClass. Much like you create controllers and services, you can create your own directives for AngularJS to use. When AngularJS [bootstraps](https://docs.angularjs.org/guide/bootstrap) your application, the [HTML compiler](https://docs.angularjs.org/guide/compiler) traverses the DOM matching directives against the DOM elements.

AngularJS directives are extended HTML attributes with the prefix ng-.

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.

**ng-bind:** Binds the content of an HTML element to a specified model property.

**ng-repeat:** Iterates over a collection (array or object) and repeats a set of HTML elements for each item in the collection.

**ng-if:** Conditionally includes or excludes a section of HTML based on an expression.

**ng-include:** Includes external HTML templates into the current template.

1. What is data binding in AngularJS?

Data binding in AngularJS is the synchronization between the model and the view.

When data in the model changes, the view reflects the change, and when data in the view changes, the model is updated as well. This happens immediately and automatically, which makes sure that the model and the view is updated at all times.

There are two main types of data binding in AngularJS:

One-way Data Binding:

1. Interpolation ({{ expression }}): You can use double curly braces {{ }} to bind expressions directly in the HTML. The expression is evaluated, and its result is inserted into the HTML at that point
2. Binding Attributes (ng-bind): The ng-bind directive is used to bind the content of a

HTML element to a model property. It is an alternative to using double curly braces for binding.

Two-way Data Binding:

* ng-model: This directive is used to create a two-way binding between an input element (like text input, textarea, or select) and a model property. Any changes in the input field are reflected in the model, and vice versa.

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

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

<h1>{{firstname}}</h1>

</div>

<script>

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

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

$scope.firstname = "John";

$scope.lastname = "Doe";

});

</script>

3. How is form validation done in angularJS

* Form validation is an essential aspect of web development, ensuring that user input meets specified criteria before being submitted to the server. AngularJS provides robust support for form validation through built-in directives and validation-related attributes, making it easy to implement client-side validation logic in AngularJS applications.
* AngularJS's form validation mechanism revolves around the use of directives and attributes to define validation rules and error messages for form elements. The ng-model directive is at the heart of form validation in AngularJS, as it binds form controls to properties on the AngularJS scope and enables the tracking of their validation status.
* To perform form validation in AngularJS, developers can use a combination of built-in directives such as **ng-required**, **ng-minlength**, **ng-maxlength**, **ng-pattern**, and ng-validators, along with validation-related attributes like required, minlength, maxlength, pattern, and **ng-messages**. These directives and attributes allow developers to specify validation rules for form controls, such as required fields, minimum and maximum lengths, and custom validation patterns.
* AngularJS also provides a mechanism for displaying validation error messages to users using the ng-messages directive. This directive enables developers to define error messages for specific validation errors and display them dynamically based on the state of the form controls.

In addition to built-in validation directives and attributes, AngularJS allows developers to define custom validation rules using the **$validators** and **$asyncValidators** properties of the ngModelController. This gives developers the flexibility to implement complex validation logic tailored to the specific requirements of their applications.

## Form State and Input State

AngularJS is constantly updating the state of both the form and the input fields.

Input fields have the following states:

* $untouched The field has not been touched yet
* $touched The field has been touched
* $pristine The field has not been modified yet
* $dirty The field has been modified
* $invalid The field content is not valid
* $valid The field content is valid

They are all properties of the input field, and are either true or false.

Forms have the following states:

* $pristine No fields have been modified yet
* $dirty One or more have been modified
* $invalid The form content is not valid
* $valid The form content is valid
* $submitted The form is submitted

They are all properties of the form, and are either true or false.

4. What is the use of AngularJS Controllers in the application?

* AngularJS controllers play a crucial role in structuring and organizing the business logic of an application. They act as intermediaries between the data model and the view, facilitating communication and data manipulation between the two components. Controllers are responsible for initializing the data model, exposing properties and methods to the view, handling user interactions, and updating the model based on user input.
* One of the primary functions of AngularJS controllers is to initialize the data model with default values. Controllers define the initial state of the application by setting up the properties and initial values of the scope object. This allows the view to render correctly when the application first loads, providing users with a consistent and predictable user interface.
* Controllers also expose properties and methods to the view, allowing it to interact with the underlying data model. By attaching properties and methods to the scope object, controllers make data and behavior accessible to the view, enabling dynamic rendering and user interaction. This separation of concerns between the controller and the view promotes code reusability and maintainability, as it allows developers to encapsulate application logic within the controller and keep the view clean and declarative.
* Furthermore, AngularJS controllers handle user interactions such as button clicks, form submissions, and input events. Controllers define event handlers that respond to user actions and update the model accordingly. For example, a controller might define a function to be executed when a button is clicked, which updates the model or performs some other action based on the user's input.
* In summary, AngularJS controllers play a vital role in structuring and organizing the business logic of an application. They initialize the data model, expose properties and methods to the view, handle user interactions, and update the model based on user input. By encapsulating application logic within controllers, developers can create modular, maintainable, and scalable AngularJS applications that provide a rich and interactive user experience.
* The [ng-controller directive](https://www.geeksforgeeks.org/angularjs-ng-controller-directive/) defines the application controller. In AngularJS, a controller is defined by a Javascript constructor function, which is used in AngularJS scope and also the function [$scope](https://www.geeksforgeeks.org/angularjs-scope/)) is defined when the controller is defining and it returns the concatenation of the $scope.firstname and $scope.lastname.

5. What is the use of AngularJS Filters in the application?

AngularJS filters are essential for manipulating and formatting data displayed in the user interface. They provide a convenient way to transform data before rendering it in the view, allowing developers to apply various formatting rules and transformations to improve the presentation and usability of their AngularJS applications.

One of the primary use cases of AngularJS filters is data formatting. Filters can be used to format data such as dates, numbers, and strings according to specific patterns and conventions. For example, the built-in date filter can format a date object into a human-readable string representation, while the number filter can format a numerical value with a specified precision and decimal separator.

Filters also enable developers to perform data manipulation and transformation tasks such as sorting, filtering, and pagination. The orderBy filter, for instance, can sort an array of objects based on a specified property, while the filter filter can filter an array based on a given predicate function or search query. These filters allow developers to dynamically manipulate data in the view without modifying the underlying data model, enhancing the flexibility and interactivity of AngularJS applications.

Additionally, AngularJS filters can be used to implement custom formatting and transformation logic tailored to the specific requirements of an application. Developers can define custom filters by registering filter functions with the AngularJS module system, making them available for use in templates and views. Custom filters can encapsulate complex formatting rules and data transformations, promoting code reuse and maintainability.

Furthermore, filters play a crucial role in enhancing the usability and accessibility of AngularJS applications by providing users with a consistent and intuitive user interface. By applying filters to data displayed in the view, developers can ensure that it is presented in a clear and understandable format, making it easier for users to interpret and interact with.

In summary, AngularJS filters are essential for manipulating and formatting data in AngularJS applications. They enable developers to apply various formatting rules and transformations to data before rendering it in the view, enhancing the presentation and usability of the application. Whether it's formatting dates, numbers, or strings, performing data manipulation tasks, or implementing custom formatting logic, filters play a vital role in creating rich and interactive user experiences in AngularJS applications.

**Demonstration of One-Way Data Binding :-**

**Input :-**   
<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

<title>One-Way Data Binding</title>

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

</head>

<body>

<div ng-controller="myCtrl">

<h1>{{message}}</h1> <!-- Data bound from the model (controller) to the view -->

</div>

<script>

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

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

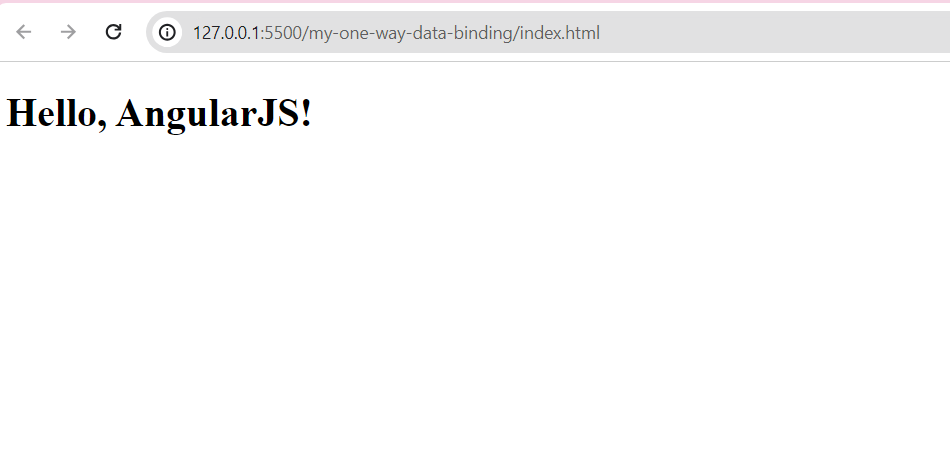
$scope.message = "Hello, AngularJS!"; // Data model (controller)

});

</script>

</body>

</html>

**Output:- **

**Demonstration of Two-Way Data Binding :-**

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

<title>Two-Way Data Binding</title>

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

</head>

<body>

<div ng-controller="myCtrl">

<input type="text" ng-model="name"> <!-- Two-way data binding -->

<h1>Hello, {{name}}!</h1> <!-- Data bound from the model (controller) to the view -->

</div>

<script>

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

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

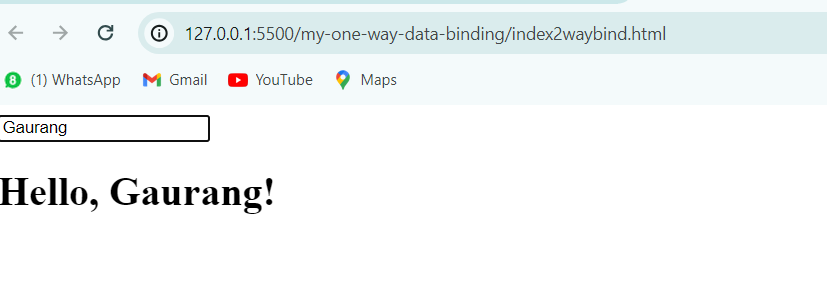
$scope.name = "AngularJS"; // Data model (controller)

});

</script>

</body>

</html>

Output:- 

**Demonstration of Form Validation :-**

Input:-

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

<title>Complex Form Validation</title>

<style>

.error-message {

color: red;

}

</style>

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

</head>

<body>

<div ng-controller="myCtrl">

<form name="registrationForm" ng-submit="submitForm()" novalidate>

<div>

<label for="username">Username:</label>

<input type="text" id="username" name="username" ng-model="user.username" ng-pattern="/^[a-zA-Z0-9\_]\*$/" required>

<span class="error-message" ng-show="registrationForm.username.$error.required && registrationForm.username.$dirty">Username is required.</span>

<span class="error-message" ng-show="registrationForm.username.$error.pattern && registrationForm.username.$dirty">Username can only contain letters, numbers, and underscores.</span>

</div>

<div>

<label for="email">Email:</label>

<input type="email" id="email" name="email" ng-model="user.email" ng-pattern="/^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/i" required>

<span class="error-message" ng-show="registrationForm.email.$error.required && registrationForm.email.$dirty">Email is required.</span>

<span class="error-message" ng-show="registrationForm.email.$error.pattern && registrationForm.email.$dirty">Please enter a valid email address.</span>

</div>

<div>

<label for="password">Password:</label>

<input type="password" id="password" name="password" ng-model="user.password" ng-minlength="6" required>

<span class="error-message" ng-show="registrationForm.password.$error.required && registrationForm.password.$dirty">Password is required.</span>

<span class="error-message" ng-show="registrationForm.password.$error.minlength && registrationForm.password.$dirty">Password must be at least 6 characters long.</span>

</div>

<div>

<label for="confirmPassword">Confirm Password:</label>

<input type="password" id="confirmPassword" name="confirmPassword" ng-model="confirmPassword" ng-pattern="user.password" required>

<span class="error-message" ng-show="registrationForm.confirmPassword.$error.required && registrationForm.confirmPassword.$dirty">Please confirm your password.</span>

<span class="error-message" ng-show="registrationForm.confirmPassword.$error.pattern && registrationForm.confirmPassword.$dirty">Passwords do not match.</span>

</div>

<button type="submit" ng-disabled="registrationForm.$invalid">Register</button>

</form>

</div>

<script>

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

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

$scope.submitForm = function() {

if ($scope.registrationForm.$valid) {

alert('Registration successful!');

// Additional form submission logic here

}

};

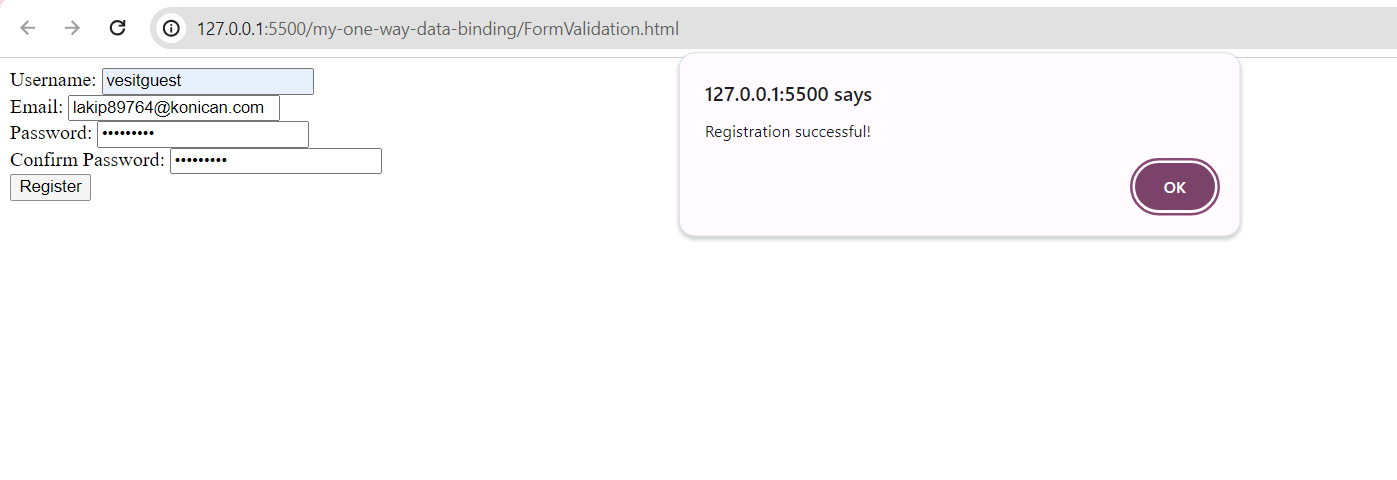
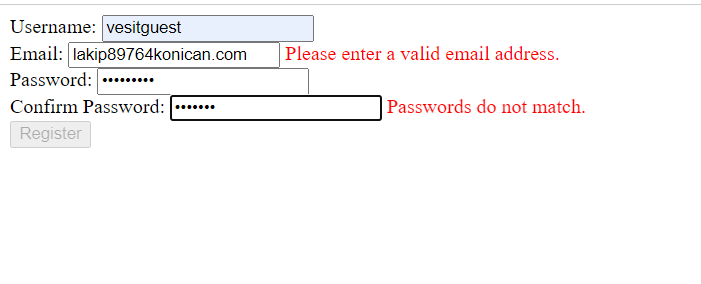
});

</script>

</body>

</html>

Output:-



**Demonstration of Filters :-**

Input:-

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

<title>AngularJS Filters</title>

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

</head>

<body>

<div ng-controller="myCtrl">

<label for="inputField">Enter values (comma-separated):</label>

<input type="text" id="inputField" ng-model="inputValues">

<button ng-click="updateItems()">Submit</button>

<ul>

<li ng-repeat="item in filteredItems">{{ item }}</li>

</ul>

</div>

<script>

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

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

$scope.inputValues = '';

$scope.items = [];

$scope.updateItems = function() {

// Split the input string into an array of values

var values = $scope.inputValues.split(',');

// Filter values containing 'a' and capitalize the initial letter if not already capitalized

$scope.filteredItems = values.filter(function(value) {

return value.toLowerCase().includes('a');

}).map(function(value) {

if (value.charAt(0).toUpperCase() !== value.charAt(0)) {

return value.charAt(0).toUpperCase() + value.slice(1);

} else {

return value;

}

});

};

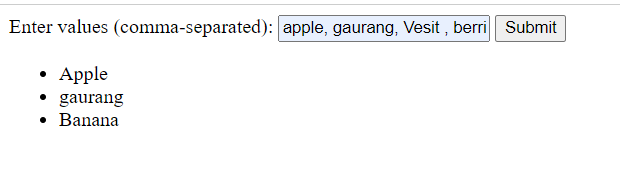
});

</script>

</body>

</html>

Output:-



Only values that contain A in it are displayed.