

MODULE 5

Services in AngularJS:

In AngularJS, services are singleton objects that provide reusable logic and functionality across an application. They help in maintaining a clean separation of concerns and avoid code duplication. Services are typically used for:

- **Sharing data** between controllers and components
- **Handling API requests** (using \$http)
- **Business logic processing**
- **Utility functions** (like date formatting, logging, etc.)

Types of Services

AngularJS provides several ways to create services:

1. Factory (.factory())
2. Service (.service())
3. Provider (.provider())
4. Value (.value())
5. Constant (.constant())

1. Factory in AngularJS (.factory())

A **factory** in AngularJS is a function that returns an object, which contains properties and methods. It is the most common way to create services because it provides flexibility and allows us to define reusable business logic.

Example: Math Service (Factory)

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

```
app.factory('MathService', function() {  
  var mathFactory = {};  
  mathFactory.square = function(x) {  
    return x * x;  
  };  
  mathFactory.cube = function(x) {  
    return x * x * x;  
  };  
  
  return mathFactory;  
});
```

- Here, MathService returns an object (mathFactory) with methods (square and cube).
- This object can be injected into controllers, directives, or other services.

2. Service in AngularJS (.service())

In AngularJS, a `.service()` is a constructor function that uses `this` to define properties and methods. Unlike a `.factory()`, which returns an object, a service is instantiated with the `new` keyword behind the scenes.

Example: Math Service (Service)

```
var app = angular.module('myApp', []);
app.service('MathService', function() {
  this.square = function(x) {
    return x * x;
  };
  this.cube = function(x) {
    return x * x * x;
  };
});
```

- The MathService uses `this` to attach methods (`square()` and `cube()`).
- It acts like a class and AngularJS creates a singleton instance of it.

3. Provider (.provider())

A provider is the most flexible way to create a service in AngularJS. Unlike factory and service, a provider allows configuration before the application runs, making it useful for dynamic values like API endpoints or authentication settings.

Example: Greeting Service (Provider)

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

app.provider('GreetService', function() {
  var greeting = "Hello"; // Default value

  this.setGreeting = function(newGreeting) {
    greeting = newGreeting;
  };

  this.$get = function() {
    return {
      sayHello: function(name) {
        return greeting + ", " + name;
      }
    };
  };
});
```

- `setGreeting(newGreeting)` allows us to configure the greeting message before the app runs.
- `$get()` returns the actual service that will be injected into controllers.

4. Value (.value())

In AngularJS, .value() is used to store simple values like strings, numbers, arrays, or objects that can be injected into controllers, services, or directives.

Example: Storing App Name

```
var app = angular.module('myApp', []);  
app.value('appName', 'My Angular App');
```

5. Constant (.constant())

In AngularJS, .constant() is used to define unchangeable values that remain the same throughout the application. Unlike .value(), constants can be injected into .config() and .run() blocks, making them useful for configuration settings like API URLs or environment variables.

The .constant() method is used for storing global settings such as API URLs, app version, or environment modes.

Example: Defining App Constants

```
var app = angular.module('myApp', []);  
  
app.constant('APP_CONFIG', {  
  appName: "My Angular App",  
  version: "1.0.0",  
  apiUrl: "https://jsonplaceholder.typicode.com"  
});
```

Tables:

The data in tables are basically repeatable, so you can use ng-repeat directives to create tables easily. The ng-if, and ng-class are used to dynamically generate rows and columns based on data.

Example

```
<html lang="en" ng-app="myApp">  
<head>  
  <title>AngularJS Table Example</title>  
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>  
</head>  
<body ng-controller="TableController">  
  
  <h2>Employee Table</h2>  
  
  <table border="1">  
    <tbody>  
      <tr ng-repeat="emp in employees">  
        <td>{{ emp.id }}</td>  
        <td>{{ emp.name }}</td>  
        <td>{{ emp.position }}</td>  
        <td>{{ emp.salary | currency:"₹" }}</td>
```

```

    </tr>
  </tbody>
</table>

```

```
</body>
```

```
<script>
```

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

```
  app.controller("TableController", function ($scope) {
```

```
    $scope.employees = [
```

```
      { id: 1, name: "Arun", position: "Manager", salary: 50000 },
```

```
      { id: 2, name: "Bharath", position: "Developer", salary: 60000 },
```

```
      { id: 3, name: "Charan", position: "Designer", salary: 45000 },
```

```
      { id: 4, name: "Dayananda", position: "Marketing", salary: 40000 }
    ];
```

```
  });
```

```
</script>
```

```
</html>
```

Employee Table

1	Arun	Manager	₹50,000.00
2	Bharath	Developer	₹60,000.00
3	Charan	Designer	₹45,000.00
4	Dayananda	Marketing	₹40,000.00

Select box

In AngularJS, you can create a select (dropdown) box using the ng-model directive to bind the selected value and ng-options or ng-repeat to populate options dynamically.

Ex:

```
<select ng-model="selectedEmployee" ng-options="emp.name for emp in employees">
```

```
  <option value="">-- Select Employee --</option>
```

```
</select>
```

```
<p>Selected Employee: {{ selectedEmployee }}</p>
```

```
<html>
```

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

```
<body>
```

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

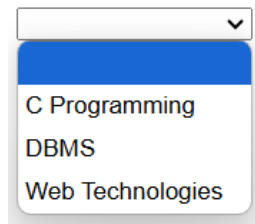
```
<select ng-model="selectedName" ng-options="x for x in names">
```

```
</select>
```

```
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.names = ["C Programming", "DBMS", "Web Technologies"];
});
</script>

</body>
</html>
```



Forms:

Forms in AngularJS are an essential part of building interactive web applications. It handles user input, validation, and submission efficiently.

AngularJS provides form handling features such as validation, two-way data binding, and form state tracking.

A form can be consisting of the many numbers of controls

1. Input field
2. Checkbox
3. Radiobox
4. Button
5. SelectBox(Dropdowns)

1) Input fields:

Input Type	Description	Example
Text	Captures user text.	<input type="text" ng-model="user.name">
Email	Requires a valid email format.	<input type="email" ng-model="user.email">
Password	Used for sensitive data.	<input type="password" ng-model="user.password">
Number	Allows numeric values.	<input type="number" ng-model="user.age">
Date	Selects a date.	<input type="date" ng-model="user.birthdate">

Checkbox	Select multiple option	<code><input type="checkbox" ng-model="user.terms"></code>
Radio	Select one option from multiple.	<code><input type="radio" ng-model="user.gender" value="Male"></code>
Select Dropdown	Choose from a list.	<code><select ng-model="user.country"><option value="India">India</option></select></code>
Textarea	Multi-line text input.	<code><textarea ng-model="user.comments"></textarea></code>

1. Text Input Field:

The `<input type="text">` field in AngularJS allows users to enter text. It works with `ng-model` for two-way data binding, meaning any change in the input field updates the model and vice versa.

Ex:

```
<input type="text" ng-model="user.name">
<p>You entered: {{ user.name }}</p>
```

2. Email Input:

The entered value follows a valid email format. `type="email"` automatically validates email format. AngularJS provides **built-in validation** using `type="email"`.

Ex:

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

3. Password Input:

The `<input type="password">` field in AngularJS is used for entering secure data. Use `type="password"` for hidden input. You can apply validation rules such as minimum length, maximum length.

Ex:

```
<input type="password" name="userPassword" ng-model="user.password" required />
```

4. Number Input:

The `<input type="number">` field in AngularJS allows users to enter numeric values. `type="number"` ensures that only numeric values can be entered. It supports validation rules like minimum and maximum values, step values, and custom patterns for more control.

```
<input type="number" name="userNumber" ng-model="user.number" required />
```

5. Date Input:

The `<input type="date">` field in AngularJS allows users to select a date from a calendar picker. `type="date"` provides a **calendar picker** for easy selection.

```
<input type="date" name="userDate" ng-model="user.date" required />
```

6. Checkboxes Input:

You can allow users to select multiple options and store them in an object. Each checkbox is bound to a different property in user.options. When checked, the value becomes true, otherwise, it's false.

```
<input type="checkbox" ng-model="user.options.newsletter" />
```

7. Radio Input:

Radio buttons in AngularJS (<input type="radio">) allow users to select only one option from a set of choices. They are commonly used for selecting gender, payment methods, survey answers, etc. ng-model="user.gender" binds the selected value to user.gender. The value="Male", value="Female", etc., define possible options.

Ex:

```
<form name="radioForm" novalidate>

  <label>

    <input type="radio" name="gender" ng-model="user.gender" value="Male" required />

    Male

  </label>

  <label>

    <input type="radio" name="gender" ng-model="user.gender" value="Female" required />

    Female

  </label>
```

8. Dropdown (Select) Input:

Dropdowns in AngularJS (<select>) allow users to choose one option from a list. They are commonly used for country selection, categories, filtering options, etc. <select> creates the dropdown list. ng-model="user.country" binds the selected value.

Ex:

```
<form name="dropdownForm" novalidate>

  <label>Select a Country:</label>

  <select name="country" ng-model="user.country" required>

    <option value="">-- Select --</option>

    <option value="USA">USA</option>

    <option value="Canada">Canada</option>

    <option value="UK">UK</option>

  </select>
```

9. Textarea Input:

A `<textarea>` in AngularJS allows users to enter multi-line text input, making it useful for comments, descriptions, feedback forms, and more. `<textarea>` creates a multi-line text box. `ng-model="user.message"` binds the user input to the message variable.

```
<label>Enter Your Message:</label>
```

```
<textarea name="message" ng-model="user.message" required></textarea>
```

```
<html lang="en" ng-app="formApp">
<head>
  <title> Form Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="FormController">

  <h2>User Registration Form</h2>

  <form name="userForm" novalidate>

    <!-- Text Input -->
    <label>Full Name:</label>
    <input type="text" name="fullName" ng-model="user.fullName" required />
    <span class="error" ng-show="userForm.fullName.$touched &&
                                userForm.fullName.$error.required">
      Name is required.
    </span>
    <br>
    <!-- Email Input -->
    <label>Email:</label>
    <input type="email" name="email" ng-model="user.email" required />
    <span class="error" ng-show="userForm.email.$touched && userForm.email.$error.required">
      Valid email is required.
    </span>
    <br>
    <!-- Password Input -->
    <label>Password:</label>
    <input type="password" name="password" ng-model="user.password" ng-minlength="6" required
    />
    <span class="error" ng-show="userForm.password.$touched &&
userForm.password.$error.minlength">
      Password must be at least 6 characters long.
    </span>
    <br>
    <!-- Number Input -->
    <label>Age:</label>
    <input type="number" name="age" ng-model="user.age" min="18" max="100" required />
    <span class="error" ng-show="userForm.age.$touched && userForm.age.$error.min">
      Minimum age is 18.
```



```

    </span>
<br>
    <!-- Date Input -->
    <label>Date of Birth:</label>
    <input type="date" name="dob" ng-model="user.dob" required />
<br>
    <!-- Checkbox -->

<br>
    <!-- Radio Buttons -->
    <label>Gender:</label>
    <label><input type="radio" name="gender" ng-model="user.gender" value="Male" required />
Male</label>
    <label><input type="radio" name="gender" ng-model="user.gender" value="Female" required />
Female</label>
    <label><input type="radio" name="gender" ng-model="user.gender" value="Other" required />
Other</label>

    <span class="error" ng-show="userForm.gender.$touched && userForm.gender.$error.required">
        Please select gender.
    </span>
<br>
    <!-- Dropdown (Select) -->
    <label>Country:</label>
    <select name="country" ng-model="user.country" required>
        <option value="">-- Select --</option>
        <option value="USA">USA</option>
        <option value="Canada">Canada</option>
        <option value="UK">UK</option>
    </select>

    <span class="error" ng-show="userForm.country.$touched &&
userForm.country.$error.required">
        Please select a country.
    </span>
<br>
    <!-- Textarea -->
    <label>Bio:</label>
    <textarea name="bio" ng-model="user.bio" ng-maxlength="200"></textarea>
    <p>{{ user.bio.length || 0 }}/200 characters</p>
<br>
    <label>
        <input type="checkbox" name="terms" ng-model="user.terms" required />
        I agree to the Terms & Conditions
    </label>
    <span class="error" ng-show="userForm.terms.$touched && userForm.terms.$error.required">
        You must agree before submitting.
    </span>
<br>

```

```
<button type="submit" ng-disabled="userForm.$invalid" ng-
click="submitForm()">Submit</button>
```

```
</form>
```

```
<!-- Display Entered Data -->
```

```
<div class="output" ng-if="submitted">
```

```
  <h3>Entered Details:</h3>
```

```
  <p><strong>Name:</strong> {{ user.fullName }}</p>
```

```
  <p><strong>Email:</strong> {{ user.email }}</p>
```

```
  <p><strong>Age:</strong> {{ user.age }}</p>
```

```
  <p><strong>Date of Birth:</strong> {{ user.dob | date:'MM/dd/yyyy' }}</p>
```

```
  <p><strong>Gender:</strong> {{ user.gender }}</p>
```

```
  <p><strong>Country:</strong> {{ user.country }}</p>
```

```
  <p><strong>Bio:</strong> {{ user.bio }}</p>
```

```
</div>
```

```
<script>
```

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

```
  app.controller("FormController", function ($scope) {
```

```
    $scope.user = {}; // Initialize empty user object
```

```
    $scope.submitted = false;
```

```
    $scope.submitForm = function () {
```

```
      if ($scope.userForm.$valid) {
```

```
        $scope.submitted = true;
```

```
      }
```

```
    };
```

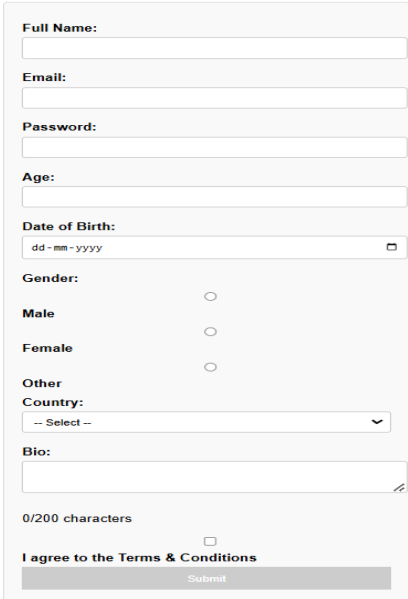
```
  });
```

```
</script>
```

```
</body>
```

```
</html>
```

User Registration Form



The image shows a user registration form with the following fields and controls:

- Full Name:** A text input field.
- Email:** A text input field.
- Password:** A text input field.
- Age:** A text input field.
- Date of Birth:** A date picker with a dropdown arrow.
- Gender:** Three radio buttons labeled "Male", "Female", and "Other".
- Country:** A dropdown menu with "-- Select --" as the placeholder.
- Bio:** A text area with a character count "0/200 characters" and a "Submit" button.
- Agree to the Terms & Conditions:** A checkbox.

Events:

Events allow to bind user interactions (such as clicks, key presses, or mouse movements) to functions in the controller or directive. AngularJS provides event directives that help manage these interactions efficiently.

AngularJS provides several built-in directives for handling events:

Event Directive	Description
ng-click	Triggers when an element is clicked.
ng-dblclick	Triggers when an element is double-clicked.
ng-mousedown	Triggers when a mouse button is pressed down.
ng-mouseup	Triggers when a mouse button is released.
ng-keydown	Triggers when a key is pressed down.
ng-keyup	Triggers when a key is released.
ng-focus	Triggers when an input field gains focus.
ng-submit	Triggers when a form is submitted.

1. ng-click:

- ng-click is an AngularJS directive that executes an expression or function when an element is clicked.
- It is commonly used for handling button clicks, toggling visibility, changing data, etc.

Syntax

```
<button ng-click="functionName()">Click Me</button>
```

When the button is clicked, the function `functionName()` is executed.

```
<html lang="en" ng-app="myApp">
<head>
  <title>ng-click Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="MainCtrl">

  <button ng-click="sayHello()">Click Me</button>
  <p>{{ message }}</p>

  <script>
    var app = angular.module('myApp', []);
    app.controller('MainCtrl', function ($scope) {
      $scope.message = "Click the button!";
    });
  </script>
</body>
</html>
```

```

    $scope.sayHello = function () {
        $scope.message = "Hello, AngularJS!";
    };
  });
</script>
</body>
</html>

```

2. ng-dblclick

- ng-dblclick is an AngularJS directive that executes a function when an element is double-clicked.
- It is useful for triggering special actions that require a double-click, such as editing a field, deleting an item with confirmation, or toggling UI elements.

Syntax

```
<button ng-dblclick="functionName()">Double Click Me</button>
```

```

<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
  <title>ng-dblclick Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="MainCtrl">

  <p ng-dblclick="changeMessage()">{{ message }}</p>

  <script>
    var app = angular.module('myApp', []);
    app.controller('MainCtrl', function ($scope) {
      $scope.message = "Double-click to change this text!";
      $scope.changeMessage = function () {
        $scope.message = "You double-clicked!";
      };
    });
  </script>
</body>
</html>

```

3. ng-mousedown

- ng-mousedown is an AngularJS directive that executes a function when the mouse button is pressed down on an element.
- It is useful for dragging effects, animations, highlighting elements, or triggering actions before releasing the mouse.

Syntax

```
<button ng-mousedown="functionName()">Press Mouse Down</button>
```

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
  <title>ng-mousedown Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="MainCtrl">

  <p ng-mousedown="changeMessage()">{{ message }}</p>

  <script>
    var app = angular.module('myApp', []);
    app.controller('MainCtrl', function ($scope) {
      $scope.message = "Press and hold mouse on this text!";
      $scope.changeMessage = function () {
        $scope.message = "Mouse button is being held!";
      };
    });
  </script>
</body>
</html>
```

4. ng-mouseup

- ng-mouseup is an AngularJS directive that executes a function when the mouse button is released after being pressed down on an element.
- It is commonly used in drag-and-drop actions, stopping animations, resetting values, or completing an action after a click.

Syntax

```
<button ng-mouseup="functionName()">Release Mouse</button>
```

5. ng-keydown

- ng-keydown is an AngularJS directive that executes a function when a key is pressed down on an input field or any other element.
- It is useful for capturing keystrokes, triggering real-time actions, form validation, and handling shortcuts.

Syntax

```
<input type="text" ng-keydown="functionName($event)" />
```

6. ng-submit

- ng-submit is an AngularJS directive used to handle form submissions.
- It prevents the default form submission behavior and executes an AngularJS function instead.
- Typically used with ng-model for form input handling.

Syntax

```
<form ng-submit="functionName()">
  <input type="text" ng-model="userInput" />
  <button type="submit">Submit</button>
</form>
```

```
<html lang="en" ng-app="myApp">
<head>
  <title>ng-submit Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="MainCtrl">
  <form ng-submit="submitForm()">
    <input type="text" ng-model="name" placeholder="Enter your name" required />
    <button type="submit">Submit</button>
  </form>
  <p>{{ message }}</p>
  <script>
    var app = angular.module('myApp', []);
    app.controller('MainCtrl', function ($scope) {
      $scope.message = "";
      $scope.submitForm = function () {
        $scope.message = "Hello, " + $scope.name + "!";
      };
    });
  </script> </body> </html>
```

Validations

AngularJS performs form validation on the client side. AngularJS monitors the state of the form and input fields (input, text-area, select), and notify the user about the current state. AngularJS also holds information about whether the input fields have been touched, modified, or not.

Form input fields have the following states:

1. **\$untouched:** It shows that field has not been touched yet.
2. **\$touched:** It shows that field has been touched.
3. **\$pristine:** It represents that the field has not been modified yet.
4. **\$dirty:** It illustrates that the field has been modified.
5. **\$invalid:** It specifies that the field content is not valid.
6. **\$valid:** It specifies that the field content is valid.
7. **\$pristine:** It represents that the fields have not been modified yet.
8. **\$submitted:** It specifies that the form is submitted.

1. \$untouched:

- \$untouched is a built-in property in AngularJS form validation.
- It indicates whether an input field has been interacted with.
- A field remains \$untouched = true until the user clicks inside it.
- Once the user clicks and then leaves the field, it becomes \$touched instead.

2. \$touched:

- \$touched is a built-in AngularJS property for form validation.
- It becomes true when the user clicks inside an input field and then leaves it.
- Used to show validation errors only after the user interacts with the field.

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
  <title>$untouched Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .error { color: red; }
  </style>
</head>
<body ng-controller="MainCtrl">
  <form name="myForm">
```

```

<input type="text" name="username" ng-model="user.name" required placeholder="Enter your
name" />
<span class="error" ng-show="myForm.username.$untouched">
  This field has not been touched yet.
</span>
<span class="error" ng-show="myForm.username.$touched &&
myForm.username.$error.required">
  Name is required!
</span>
</form>
<script>
  var app = angular.module('myApp', []);
  app.controller('MainCtrl', function ($scope) {
    $scope.user = {};
  });
</script>
</body>
</html>

```

3. \$dirty

- \$dirty is a built-in property in AngularJS form validation.
- It becomes true when the user changes the input value.
- If the field remains unchanged, \$dirty = false.
- It is used to track modified fields in a form.

```

<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
  <title>$dirty Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .error { color: red; }
    .success { color: green; }
  </style>
</head>
<body ng-controller="MainCtrl">

  <form name="myForm">

```



```
<input type="text" name="username" ng-model="user.name" required placeholder="Enter
your name" />
```

```
<span class="error" ng-show="myForm.username.$dirty &&
myForm.username.$error.required">
```

```
    Name is required!
```

```
</span>
```

```
<span class="success" ng-show="myForm.username.$dirty &&
!myForm.username.$error.required">
```

```
    Looks good!
```

```
</span>
```

```
</form>
```

```
<script>
```

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

```
    app.controller('MainCtrl', function ($scope) {
```

```
        $scope.user = {};
```

```
    });
```

```
</script>
```

```
</body>
```

```
</html>
```

- Initially, the field is clean (\$dirty = false), so no messages are shown.
- If the user types something and then deletes it, \$dirty = true, and the "Name is required!" message appears.
- If the user enters a value, a "Looks good!" message appears.

4. \$invalid:

- \$invalid is a built-in property in AngularJS form validation.
- It becomes true when the input field fails validation (e.g., required field left blank, incorrect email format, etc.).
- If the input is valid, \$invalid = false.
- Works together with \$error to detect specific validation issues.

5. \$valid:

- \$valid is a built-in AngularJS property used in form validation.
- It becomes true when the input field passes all validation rules (e.g., required field is filled, valid email format, etc.).
- If any validation rule fails, \$valid = false.
- Works together with \$invalid (\$valid is opposite of \$invalid).

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
  <title>$invalid Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .error { color: red; }
  </style>
</head>
<body ng-controller="MainCtrl">

  <form name="myForm" novalidate>
    <input type="email" name="userEmail" ng-model="user.email" required placeholder="Enter your
email" />

    <span class="error" ng-show="myForm.userEmail.$touched && myForm.userEmail.$invalid">
      Invalid or missing email!
    </span>

    <br><br>
    <button type="submit" ng-disabled="myForm.userEmail.$invalid">Submit</button>
  </form>

  <script>
    var app = angular.module('myApp', []);
    app.controller('MainCtrl', function ($scope) {
      $scope.user = {};
    });
  </script>
</body>
</html>
```

- If the email field is empty or incorrect, \$invalid = true, and an error message appears.
- The submit button is disabled until the user enters a valid email.

6. \$submitted:

- \$submitted is a built-in AngularJS property used in form validation.
- It becomes true when the form has been submitted.
- Used to show validation messages only after submission.
- Helps prevent premature error messages before the user submits the form.

```

<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
  <title>$submitted Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .error { color: red; }
    .success { color: green; }
  </style>
</head>
<body ng-controller="MainCtrl">
  <form name="myForm" ng-submit="submitForm(myForm)" novalidate>
    <input type="text" name="username" ng-model="user.name" required placeholder="Enter your
name" />
    <span class="error" ng-show="myForm.$submitted && myForm.username.$error.required">
      Name is required!
    </span>
    <br><br>
    <button type="submit">Submit</button>
  </form>
  <script>
    var app = angular.module('myApp', []);
    app.controller('MainCtrl', function ($scope) {
      $scope.user = {};
      $scope.submitForm = function (form) {
        if (form.$valid) {
          alert("Form submitted successfully!");
        }
      };
    });
  </script>
</body>
</html>

```