AngularJS Forms

Topics : AngularJS
Written on January 09, 2024

AngularJS provides powerful features for working with forms, allowing you to easily manage and validate user input. Here's a guide on how to create and work with forms in AngularJS:

Basic Form:

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
<meta charset="UTF-8">
<title>AngularJS Forms</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="FormController">
<form ng-submit="submitForm()">
<label>Name:</label>
<input type="text" ng-model="formData.name" required>

<label>Email:</label>
<input type="email" ng-model="formData.email" required>

<button type="submit">Submit</button>
</form>

<script>
angular.module('myApp', []).controller('FormController', function($scope) {
$scope.formData = {};

$scope.submitForm = function() {
// Form submission logic goes here
console.log('Form submitted:', $scope.formData);
};
});
</script>
</body>
</html>
```

In this example:

- `ng-model` is used for two-way data binding between form elements and the controller.
- The `required` attribute ensures that the fields are not empty.

- `ng-submit` is used to define the function to be called when the form is submitted.

Form Validation:

AngularJS provides built-in validation directives, such as `ng-required`, `ng-minlength`, `ng-maxlength`, `ng-pattern`, and more.

```
<label>Password:</label>
<input type="password" ng-model="formData.password" ng-minlength="6" ng-maxlength="20" required>

<label>Confirm Password:</label>
<input type="password" ng-model="formData.confirmPassword" ng-minlength="6" ng-maxlength="20" required>
<span ng-show="formData.password !== formData.confirmPassword">Passwords do not match.</span>
```

In this example, the password field must have a length between 6 and 20 characters. Additionally, a message is displayed if the passwords do not match.

Custom Validation:

You can create custom validation functions in the controller to perform more complex validation.

```
<label>Custom Validation Field:</label>
<input type="text" ng-model="formData.customField" ng-pattern="/^[a-zA-Z]*$/" required>
<span ng-show="customValidation()">Only letters are allowed.</span>
```

In the controller:

```
$scope.customValidation = function() {
return !/^[a-zA-Z]*$/.test($scope.formData.customField);
};
```

This example uses a custom validation function (`customValidation`) to check if only letters are allowed in the input.

Handling Form States:

You can use properties like `$pristine`, `$dirty`, `$valid`, and `$invalid` to check the state of the form and its elements.

```
<div ng-show="myForm.$dirty && myForm.$invalid">
The form is not valid.
</div>

<div ng-show="myForm.$valid">
The form is valid!
</div>
```

Conditional Form Elements:

You can conditionally show or hide form elements based on certain criteria.

```
<label>Toggle Field:</label>
<input type="checkbox" ng-model="showField">

<div ng-show="showField">
<label>Conditional Field:</label>
<input type="text" ng-model="formData.conditionalField" required>
</div>
```

In this example, the second field is shown or hidden based on the value of the checkbox.