WT

ass5

Certainly, I'll provide explanations for the AngularJS directives (ng-attributes) used in the code:

1. `ng-app`: This directive defines the root of the AngularJS application. In this code, `ng-app="TodoApp"` specifies the application's name as "TodoApp."
2. `ng-controller`: This directive attaches a controller to a specific HTML element, defining the scope for the associated HTML content. In this code, `ng-controller="TodoController"` links the controller "TodoController" to the `<div>` element.
3. `ng-model`: This directive binds the value of an HTML element (e.g., an input field) to a property in the AngularJS scope. For example, `ng-model="newTodo"` connects the input field's value to the "newTodo" property in the controller.
4. `ng-repeat`: The `ng-repeat` directive iterates over a collection in the scope and generates HTML elements for each item in the collection. In this code, `ng-repeat="todo in todos"` repeats the `<li>` element for each "todo" item in the "todos" array.
5. `ng-submit`: This directive specifies an AngularJS expression to be executed when a form is submitted. In this code, `ng-submit="addItem()"` calls the "addItem" function when the form is submitted.
6. `ng-show` **and** `ng-hide`: These directives conditionally show or hide elements based on the evaluated expression. In the code, `ng-show="todo.editing"` and `ng-hide="todo.editing"` are used to show or hide the input field and text based on the "editing" property of the to-do item.
7. `ng-click`: The `ng-click` directive defines an expression to be executed when an element is clicked. In this code, it's used with "ng-click" to specify the functions to call when buttons are clicked, such as "editItem(todo)" and "deleteItem(todo)."

These AngularJS directives enable dynamic behavior in the application by allowing you to interact with the data and perform actions such as adding, editing, and deleting to-do items. They help synchronize the view (HTML) with the model (JavaScript) by updating the DOM elements based on the application's data and logic.

code

```
<!DOCTYPE html>
<html ng-app="TodoApp">
<head>
    <title>AngularJS To-Do List</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.3/angular.min.js"></script>
</head>
<body>
    <div ng-controller="TodoController">
        <h1>To-Do List</h1>
        <form ng-submit="addItem()">
            <!-- This is an input field where you can type in new to-do items. The ng-model directive binds the input's value to the "newTo
            <input type="text" ng-model="newTodo" placeholder="Add a new to-do" required>
            <button type="submit">Add</button>
        </form>
        <ul>
            <li ng-repeat="todo in todos">
                <span ng-hide="todo.editing">{{ todo.text }}</span>
                <!-- Input field for editing a to-do (hidden by default) -->
                <input type="text" ng-show="todo.editing" ng-model="todo.text" ng-blur="updateItem(todo)">
                <!-- Edit button to switch to edit mode -->
                <button ng-click="editItem(todo)">Edit</button>
                <!-- Delete button to remove the to-do item -->
                <button ng-click="deleteItem(todo)">Delete</button>
            </li>
        </ul>
    </div>

    <script>
        angular.module('TodoApp', [])
            .controller('TodoController', function ($scope) {
                $scope.todos = [];
                $scope.newTodo = '';

                // Function to add a new to-do item
                $scope.addItem = function () {
```

```
                    if ($scope.newTodo) {
                        $scope.todos.push({ text: $scope.newTodo, editing: false });
                        $scope.newTodo = '';
                    }
                };

                // Function to delete a to-do item
                $scope.deleteItem = function (todo) {
                    var index = $scope.todos.indexOf(todo);
                    $scope.todos.splice(index, 1);
                };

                // Function to switch to edit mode
                $scope.editItem = function (todo) {
                    todo.editing = true;
                };

                // Function to exit edit mode
                $scope.updateItem = function (todo) {
                    todo.editing = false;
                };
            });
    </script>
</body>
</html>
```

Registration form

<!DOCTYPE html>
<html>
<head>
<title>Registration Form</title>
</head>
<body>
<h1>Registration</h1>
<form id="registrationForm">
<label for="username">Username:</label>
<input type="text" id="username" name="username" required>


```
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <span id="emailError" style="color: red;"></span><br><br>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
    <span id="passwordError" style="color: red;"></span><br><br>

    <button type="button" onclick="validateForm()">Register</button>
</form>

<script>
    function validateForm() {
        // Get form elements
        var username = document.getElementById("username");
        var email = document.getElementById("email");
        var password = document.getElementById("password");

        // Get error message elements
        var usernameError = document.getElementById("usernameError");
        var emailError = document.getElementById("emailError");
        var passwordError = document.getElementById("passwordError");

        // Reset error messages
        usernameError.textContent = "";
        emailError.textContent = "";
        passwordError.textContent = "";

        // Check for empty fields
        if (username.value === "") {
            usernameError.textContent = "Username is required.";
```

```
            return;
        }

        if (email.value === "") {
            emailError.textContent = "Email is required.";
            return;
        }

        if (password.value === "") {
            passwordError.textContent = "Password is required.";
            return;
        }

        // If all fields are filled, proceed with registration
        alert("Registration successful!");
        // You can add further processing here, such as sending data to a server.
    }
</script>
```

</body>
</html>

Bootrap

<!DOCTYPE html>
<html>
<head>
<title>Bootstrap Example</title>
<!-- Include Bootstrap CSS (Bootstrap usage starts here) -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
<!-- Navigation Bar (Bootstrap classes used for styling) -->
<nav class="navbar navbar-expand-lg navbar-light bg-light">
My Website
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

</button>

<ul class="navbar-nav">
<li class="nav-item active">
Home

<li class="nav-item">
About

<li class="nav-item">
Services

<li class="nav-item">
Contact

</nav>

```
<!-- Content Container (Bootstrap classes used for layout) -->
<div class="container mt-4">
    <div class="row">
        <div class="col-md-4">
```

```
            <!-- Card Component (Bootstrap classes used for styling) -->
            <div class="card">
                <img src="<https://via.placeholder.com/150>" class="card-img-top" alt="Image">
                <div class="card-body">
                    <h5 class="card-title">Card Title</h5>
                    <p class="card-text">This is a simple card component using Bootstrap.</p>
                    <a href="#" class="btn btn-primary">Read More</a>
                </div>
            </div>
        </div>
    </div>
</div>

<!-- Include Bootstrap JS and jQuery (Bootstrap JavaScript usage starts here) -->
<script src="<https://code.jquery.com/jquery-3.5.1.slim.min.js>"></script>
<script src="<https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js>"></script>
<script src="<https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js>"></script>
```

</body>
</html>