# Session 4

# Custom Directives, Scope, and Services

# Session Overview

In this session, you will be able to:

- Outline how to create a custom directive
- Use a custom directive
- Describe the concept of scope in AngularJS
- Explain services in AngularJS

# Custom Directives

## Creating a Custom Directive

AngularJS allows us to create our own application specific, custom directives, in case the built-in directives don't work the way we require.

```
var app = angular.module('myApp', []);
app.directive('myCustomDirective',
function() {
return {
restrict: 'AE',
template: '<h3>Hello
AngularJS!!</h3>
<p>I was made inside a Custom
directive<p>'
}
```

***A sample code using custom directive***

# Custom Directives

Invoking a Custom Directive

For calling a custom directive in HTML, we can simply use it as an Element.

```
<body ng-app= “myApp”>
<my-custom-directive></my-custom-directive>
</body>
```

***A sample code using custom directive as an element***

```
<body ng-app= “myApp”>
<div my-custom-directive></div>
</body>
```

***A sample code using custom directive as an attribute***

# Custom Directives

Invoking a Custom Directive

In AngularJS, the restrict values restricts the use of custom directive as an Element or as an Attribute

The allowed restrict values are:

- E for Element name
- A for Attribute
- C for Class
- M for Comment

Where the default value is EA (Element names and attribute names can invoke the directive)

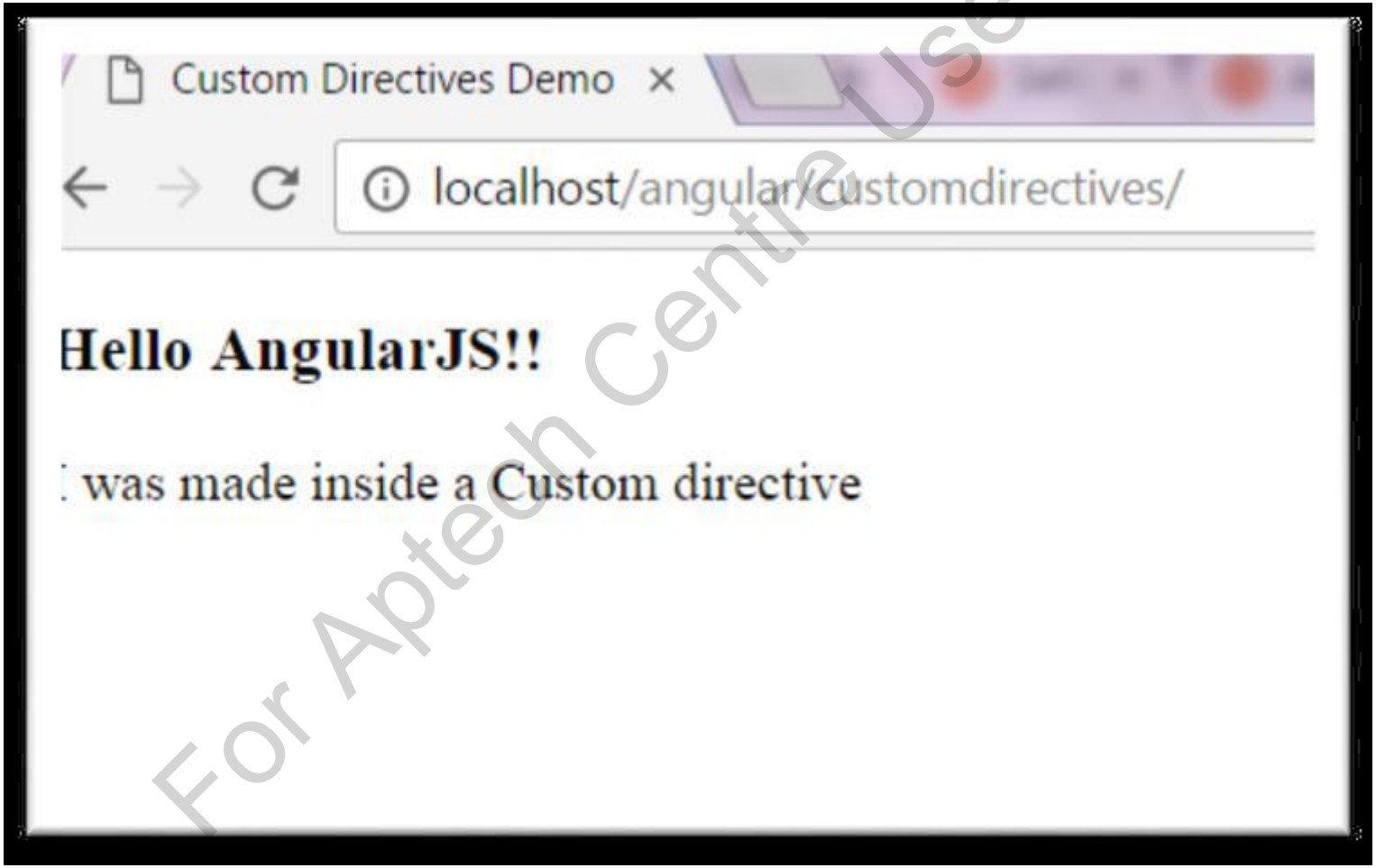
# Custom Directives

The complete code using custom directive

```
1. <!DOCTYPE html>
2. <html>
3. <script
4.src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
5. <body ng-app="myApp">
6. <w3-custom-directive></w3-custom-directive>
7. <div w3-custom-directive></div>
8. <script>
9. var app = angular.module("myApp", []);
10. app.directive('w3CustomDirective', function() {
11. return {
12. restrict: 'A',
13. template: '<h3>Hello AngularJS!!</h3><p>I was made inside a
14.Custom directive<p>'
15. };
16. });
17. </script>
18. </body>
19. </html>
```

# Custom Directives

The complete code using custom directive - Output



# Scopes

Introduction to Scopes

- The scope of AngularJS is the model.
- It is a JavaScript object with properties and methods available for both the view and the controller.
- It gives the execution context for expressions used in the application.
- The three types of scopes are:
  - Shared scope
  - Inherited scope
  - Isolated scope

# Scopes

## Scope Hierarchies

- All applications have a $rootScope which is the scope created on the HTML element that contains the ng-app directive.
- The $rootScope is available in the entire application.
- When a variable has the same name in both the current scope and in the $rootScope, the application makes use of the variable in the current scope.

# Scopes

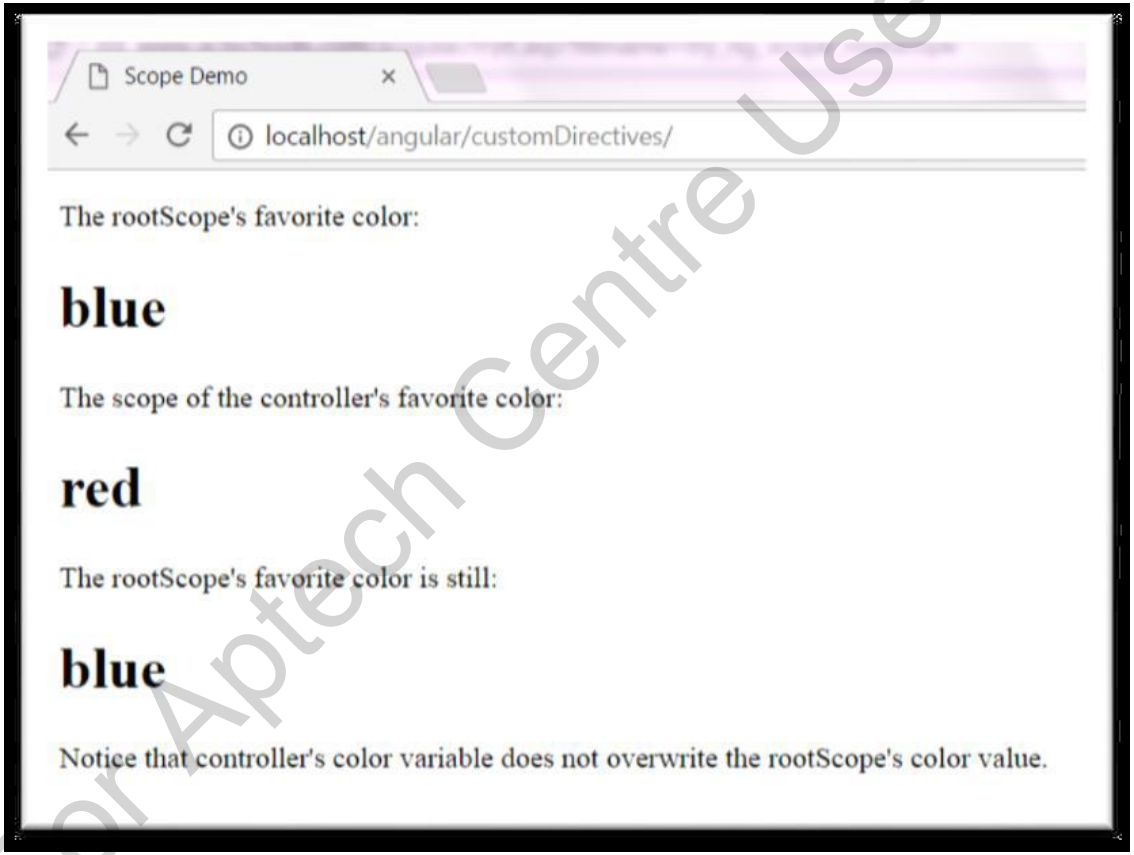
## Scope Hierarchies

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Scope Demo</title>
6. <script src="angular.min.js"></script>
7. </head>
8. <body ng-app="myApp">
9. <p>The rootScope's favorite color:</p>
10. <h1>{{color}}</h1>
11. <div ng-controller="myCtrl">
12. <p>The scope of the controller's
favorite color:</p>
13. <h1>{{color}}</h1>
14. </div>
15. <p>The rootScope's favorite color is
still:</p>
16. <h1>{{color}}</h1>
17. <script>
18. var app = angular.module('myApp', []);
19. app.run(function($rootScope) {
20. $rootScope.color = 'blue';
21. });
22. app.controller('myCtrl', function($scope)
{
23. $scope.color = "red";
24. });
24. </script>
25. <p>Notice that controller's color
variable does not overwrite the rootScope's
color value.</p>
26. </body>
27. </html>
```

***$rootScope and $scope Example- Code***

# Scopes

## Scope Hierarchies




***$rootScope and $scope Example- Output***

# Scopes

## Nested Scopes and Controllers

```
1.<!DOCTYPE html>
2.<html lang="en">
3.<head>
4.<meta charset="UTF-8">
5.<title>Nested Scope Demo</title>
6.<script src="angular.min.js"></script>
7.<script src="script.js"></script>
8.</head>
9.<body ng-app="myApp">
10.<div>
11.<h2>Nested controllers with model variables
defined directly on the scopes</h2>
(typing on an input field, with a data binding to
the model, overrides the same
variable of a parent scope)
12.</div>
13.<div ng-controller="firstControllerScope">
14.<h3>First controller</h3>
15.<strong>First name:</strong>
{{firstName}}<br />
16.<br />
17.<label>Set the first name: <input type="text" ng-
model="firstName"/></label><br />
18.<br />
19.<div ng-controller="secondControllerScope">
20.<h3>Second controller (inside First)</h3>
21.<strong>First name (from First):</strong>
{{firstName}}<br />
22.<strong>Last name (new variable):</strong>
{{lastName}}<br />
23.<strong>Full name:</strong> {{getFullName()}}<br
/>
24.<br />
25.<label>Set the first name: <input type="text" ng-
model="firstName"/></label><br />
26.<label>Set the last name: <input type="text" ng-
model="lastName"/></label><br />
27.<br />
28.<div ng-controller="thirdControllerScope">
29.<h3>Third controller (inside Second and First)</h3>
30.<strong>First name (from First):</strong>
{{firstName}}<br />
```

***Nested Scopes and Controllers Example- HTML Code***

## Nested Scopes and Controllers

```
36.<label>Set the first name: <input type="text"
ng-model="firstName"/></label><br />
37.<label>Set the middle name: <input
type="text" ng-
model="middleName"/></label><br />
38.<label>Set the last name: <input type="text"
ng-model="lastName"/></label>
39.</div>
40.</div>
41.</div>
42.<br />
43.<p>
44.<h2>Nested controllers with model variables
defined inside objects</h2>
45. (typing on an input field, with a data binding
to the model, acts on a specific
object without overriding variables)
46.</p>
47.<div ng-controller="firstControllerObj">
48.<h3>First controller</h3>
49.<strong>First name:</strong>
{{firstModelObj.firstName}}<br />
50.<br />
51.<label>Set the first name: <input type="text"
ngmodel="
firstModelObj.firstName"/></label><br />
52.<br />
53.<div ng-controller="secondControllerObj">
54.<h3>Second controller (inside First)</h3>
55.<strong>First name (from First):</strong>
{{firstModelObj.firstName}}<br />
56.<strong>Last name (from Second):</strong>
{{secondModelObj.lastName}}<br />
57.<strong>Full name:</strong> {{getFullName()}}<br
/>
58.<br />
59.<label>Set the first name: <input type="text"
ngmodel="
firstModelObj.firstName"/></label><br />
60.<label>Set the last name: <input type="text"
ngmodel="
secondModelObj.lastName"/></label><br />
>
```

***Nested Scopes and Controllers Example- HTML Code***

# Scopes

## Nested Scopes and Controllers

```
61.<br />
62.<div ng-controller="thirdControllerObj">
63.<h3>Third controller (inside Second and First)</h3
64.<strong>First name (from First):</strong> {{firstModelObj.firstName}}<br />
65.<strong>Middle name (from Third):</strong> {{thirdModelObj.middleName}}<br />
66.<strong>Last name (from Second):</strong> {{secondModelObj.lastName}}<br />
67.<strong>Last name (from Third):</strong> {{thirdModelObj.lastName}}<br />
68.<strong>Full name (redefined in Third):</strong> {{getFullName()}}<br />
69.<br />
70.<label>Set the first name: <input type="text" ngmodel="
firstModelObj.firstName"/></label><br />
71.<label>Set the middle name: <input type="text" ngmodel="
thirdModelObj.middleName"/></label><br />
72.<label>Set the last name: <input type="text" ngmodel="
thirdModelObj.lastName"/></label>
73.</div>
74.</div>
75.</div>
76.</body>
77.</html>
```

***Nested Scopes and Controllers Example- HTML Code***

# Scopes

## Nested Scopes and Controllers

```
1.var app = angular.module('myApp', []);
2.app.controller('firstControllerScope',
function($scope){
3. // Initialize the model variables
4. $scope.firstName = "Sachin";
5.});
6.app.controller('secondControllerScope',
function($scope){
7. // Initialize the model variables
8. $scope.lastName = "Pilot";
9. // Define utility functions
10. $scope.getFullName = function ()
11. {
12.return $scope.firstName + " " +
$scope.lastName;
13. };
14.});
15.app.controller('thirdControllerScope',
function($scope){
16. // Initialize the model variables
17. $scope.middleName = "Ramesh";
18. $scope.lastName = "Tendulkar";
19. // Define utility functions
){
20. $scope.getFullName = function ()
21. {
22.return $scope.firstName + " " + $scope.middleName
+ " " + $scope.lastName;
23. };
24.});
25.app.controller('firstControllerObj', function($scope){
26. // Initialize the model object
27. $scope.firstModelObj = {
28.firstName: "Sachin"
29. };
30.});
31.app.controller('secondControllerObj', function($scope){
32. // Initialize the model object
33. $scope.secondModelObj = {
34.lastName: "Pilot"
35. };
36. // Define utility functions
37. $scope.getFullName = function ()
```

***Nested Scopes and Controllers Example- JavaScript Code***

# Scopes

## Nested Scopes and Controllers

```
{
39.return $scope.firstModelObj.firstName + " " +
40. $scope.secondModelObj.lastName;
41. };
42.});
43.app.controller('thirdControllerObj', function($scope){
44. // Initialize the model object
45. $scope.thirdModelObj = {
46.middleName: "Ramesh",
47.lastName: "Tendulkar"
48. };
49. // Define utility functions
50. $scope.getFullName = function ()
51. {
52.return $scope.firstModelObj.firstName + " " +
53. $scope.thirdModelObj.middleName + " " +
54. $scope.thirdModelObj.lastName;
55. };
56.});
```

***Nested Scopes and Controllers Example- JavaScript Code***

# Scopes

## Nested Scopes and Controllers

Nested Scope Demo

localhost/angular/customDirectives/nested.html

**Nested controllers with model variables defined directly on the scopes**

(typing on an input field, with a data binding to the model, overrides the same variable of a parent scope)

**First controller**

**First name**: Sachin

Set the first name: Sachin

**Second controller (inside First)**

**First name (from First)**: Sachin
**Last name (new variable)**: Pilot
**Full name**: Sachin Pilot

Set the first name: Sachin
Set the last name: Pilot

**Third controller (inside Second and First)**

**First name (from First)**: Sachin
**Middle name (new variable)**: Ramesh
**Last name (from Second)**: Pilot
**Last name (redefined in Third)**: Tendulkar
**Full name (redefined in Third)**: Sachin Ramesh Tendulkar

Set the first name: Sachin
Set the middle name: Ramesh
Set the last name: Tendulkar

**Nested controllers with model variables defined inside objects**

(typing on an input field, with a data binding to the model, acts on a specific object without overriding variables)

**First controller**

**First name**: Sachin

Set the first name: Sachin

**Second controller (inside First)**

**First name (from First)**: Sachin
**Last name (from Second)**: Pilot
**Full name**: Sachin Pilot

Set the first name: Sachin
Set the last name: Pilot

**Third controller (inside Second and First)**

**First name (from First)**: Sachin
**Middle name (from Third)**: Ramesh
**Last name (from Second)**: Pilot
**Last name (from Third)**: Tendulkar
**Full name (redefined in Third)**: Sachin Ramesh Tendulkar

Set the first name: Sachin
Set the middle name: Ramesh
Set the last name: Tendulkar

***Nested Scopes and Controllers Example- Output***

# Services

Introduction to Services

- Services' refer to simple objects that does some sort of work.
- They are JavaScript functions and are responsible to do a specific task only.
- They are injected using dependency injection mechanism of AngularJS.
- Some built-in services provided by AngularJS are, $http, $route, $window and$location.

# Services

## Sample code using $http service

```
1.<!DOCTYPE html>
2.<html lang="en">
3.<head>
4. <meta charset="UTF-8">
5. <title>$http service demo</title>
6. <script src="angular.min.js"></script>
7.</head>
8.<body>
9. <div ng-app="myApp" ng-controller="myCtrl">
10. <p>Today's welcome message is:</p>
11. <h1>{{myWelcome}}</h1>
12. </div>
13. <p>The $http service requests a page on the server, and the response is set as the value of the
"myWelcome" variable.</p>
14. <script>
15. var app = angular.module('myApp', []);
16. app.controller('myCtrl', function($scope, $http) {
17. $http.get("welcome.html")
18. .then(function(response) {
19. $scope.myWelcome = response.data;
20. });
21. });
22. </script>
23.</body>
24.</html>
```

We use
http service for
reading data
from remote servers

# Services

## $http service


***$http service – Example - Output***

# Services

$location service

- The $location service has methods which return information about the location of the current Web page.
- It also keeps itself and the URL in synchronization.
- Any modification made to $location is passed to the URL.
- Whenever the URL changes (such as when a new route is loaded) the $location service updates itself.
- $location updates the browser's URL to navigate to a different route or to watch for changes in $location.

# Summary

- We create new directives using the .directive method; the arguments we provide to this are the name of the new directive and a function that creates the directive.

- We have to use the camel case convention for the name of the custom directive when we define it.

- The allowed restrict values are:
  - E for Element name
  - A for Attribute
  - C for Class
  - M for Comment

- On the view, the custom directive is used by separating the camel case name by using a hyphen/dash, to separate the words. We need to follow this strictly.

# Summary

- The $rootScope is available in the entire application.
- If a variable has the same name in both the current scope and in the $rootScope, the application uses the variable in the current scope.
- Services are JavaScript functions and are responsible to do a specific task only.
- Services are injected using dependency injection mechanism of AngularJS.