LEARN. DO. EARN

ACADGILD

FRONT END DEVELOPMENT (WITH ANGULARJS)

Website : http://www.acadgild.com
LinkedIn : https://www.linkedin.com/company/acadgild
Facebook : https://www.facebook.com/acadgild

Session 14 – AngularJS Directives

Agenda – AngularJS Directives

1. Custom Directives
2. Creating Custom Directives Steps
3. Restrict Property
4. Understanding Directive Priority
5. Using templateUrl For External Templates
6. transclude Property
7. Introduction To Scopes

Custom Directives

- Custom directives are used in AngularJS to extend the functionality of HTML.
- Custom directives are defined using **"directive"** function.
- A custom directive simply replaces the element for which it is activated.
- AngularJS provides support to create custom directives for following type of elements.
  - Element directives - Directive activates when a matching element is encountered.
  - Attribute - Directive activates when a matching attribute is encountered.
  - CSS - Directive activates when a matching css style is encountered.
  - Comment - Directive activates when a matching comment is encountered.

Creating Custom Directives Steps

Step 1: You register a directive with a module. Here is an example of how that looks:

```
myapp = angular.module("myapp", []);
myapp.directive('div', function() { var directive = {};
directive.restrict = 'E'; /* restrict this directive to elements */
directive.template = "My first directive: {{textToInsert}}";
return directive; });
```

Step 2: Imagine that your HTML page has this HTML:

```
<div ng-controller="MyController" >
<div>This div will be replaced</div>
</div>
```

Step 3: Then the added directive would be activated when AngularJS finds the inner div element. Instead of this divelement, this HTML will be inserted:

```
My first directive: {{textToInsert}}
```

Restrict Property

- The restrict property declares if the directive can be used in a template as an element, attribute, class, comment, or any combination.

"E":	Element	<my-directive>
"A":	Attribute	<div my-directive>
"C":	Class	<div class="my-directive“>
"M":	Comment:	<!-- directive: my-directive exp -->

- Combine (e.g. "EA") for more flexibility

Understanding Directive Priority

- Directives have priority property.
- In case there are multiple directives in an element, the directive with the higher priority gets applied first.
- If we have multiple directives on a single DOM element and if the order in which they're applied matters, we can use the priority property to order their application.

- **Note** that higher number priority directives run first. The default priority is 0 if we don't specify one.

- **Example:** For **ng-repeat**, we use a priority value of 1000.

Using templateUrl For External Templates

- The template property has a drawback of getting the html code in the JavaScript code.
- A better approach is to keep the html code in a separate html file.
- We can achieve this by using the **templateUrl** property.

transclude Property

- We can also move the original content within the new template through the transclude property.

- If set to true, the directive will delete the original content within the directive, but make it available for reinsertion within the template through a directive called **ng-transclude**.

Introduction to Scopes

- Scope is an object that refers to the application model.
- Scopes are arranged in hierarchical structure which mimic the DOM structure of the application.

- New scope - We can create that inherits from our enclosing controller’s scope. Here, we will have the ability to read all the values in the scopes and this scope will be shared with any other directives on our DOM element that request this kind of scope and can be used to communicate with them.

- Isolated scope – It does not any model properties from its parent. We can use this type of scope when we need to isolate the operation of our directive from the parent scope usually while creating reusable components.

Lets Discuss Assignments