CitiusTech

# Directives

May 2015

## Agenda

- **What Are Directives?**
- Matching Directives
- Inbuilt Directives
- Custom Directives
- Use of Restrict Function in Directives
- Isolating Scope of the Directive
- Dom Manipulation & Adding Event Listeners

## What Are Directives?

- Directives are HTML DOM elements, attributes or CSS class names that inform Angular’s HTML compiler to attach a specified behavior to that DOM element

- Directives have the ability to execute methods, define behavior, attach controllers and $scope objects, manipulate the DOM  etc.

- AngularJS has predefined in-built directives for use.

  **i.e. ng-bind, ng-model, ng-class**

- AngularJS also have provision to create custom directives .

- When AngularJS bootstraps your application, the HTML compiler traverses the DOM matching directives against the DOM elements.

## Agenda

- What Are Directives?
- **Matching Directives**
- Inbuilt Directives
- Custom Directives
- Use of Restrict Function in Directives
- Isolating Scope of the Directive
- Dom Manipulation & Adding Event Listeners

CitiusTech

## Matching Directives (1/2)

- AngularJS normalizes an element's tag and attribute name to determine which elements match which directives.

- AngularJS refers the directive by their case sensitive camelCase name.

  i.e. ngModel

- As HTML is not case sensitive, Angular refers the directives in the DOM by lower-case forms, by using dash-delimited attributes on DOM

  i.e. ng-model

- Below forms are all equivalent and match the **ngBind** directive

  <span **ng-bind**=“patientName"></span> (**Prefer to use**)

  <span **ng:bind**="patientName”></span>

  <span **ng_bind**="patientName"></span>

  <span **data-ng-bind**="patientName"></span>

  (**Prefer to use for HTML validating tool**)

  <span **x-ng-bind**="patientName"></span>

## Matching Directives (2/2)

- **Normalization Process**
  - Strip x- and data- from attribute or element
  - Convert : , - or _ delimited name to camelCase

- Compiler can match the directives based on element name, DOM attribute, class name as well as comments.

- Below forms are all equivalent to match the directive

```
<patient-Tagline></patient-Tagline >      (Prefer to use)
<span patient-Tagline ="exp"></span>  (Prefer to use)
<!-- directive: patient-Tagline  exp -->
<span class=" patient-Tagline : exp;"></span>
```

## Agenda

- What Are Directives?
- Matching Directives
- **Inbuilt Directives**
- Custom Directives
- Use of Restrict Function in Directives
- Isolating Scope of the Directive
- Dom Manipulation & Adding Event Listeners

## Inbuilt Directives (1/7)

- AngularJS comes with a lot of inbuilt directives which makes life easy.

**ng-model**

- This directive is used to bind the value of the input DOM element to the $scope model in the controller.

```
<input ng-model=“patientName" placeholder="Enter your name"/>
```

**ng-init**

- This directive is a function that runs at bootstrap time.
- It allows to set default variables prior to running any other functions during runtime

```
<b ng-init=‘patientName = “Test, Patient"'>
        {{patientName}}
 </b>
```

**ng-click**

- This directive registers a listener with the DOM element.
- When the DOM listener fires, Angular executes the expression and updates the view as normal

```
<button ng-click=“submit">Submit Form</button>
```

## Inbuilt Directives (2/7)

**ng-show / ng-hide**

- These directives show or hide a portion of the DOM depending on whether the expression is true or false.

```
<div ng-show="shouldShow">
        <h3>{{ patientName}}</h3>
</div>
<div ng-hide="shouldShow">
        <h3>{{ patientName}}</h3>
</div>
```

**ng-repeat**

- This directive loads a template for each item in a collection.
- Each copy of the template gets its own scope.

```
<ul>
      <li ng-repeat="patient inlistOfPatients">
                  {{patient.name}}
       </li>
</ul>
```

## Inbuilt Directives (3/7)

**Inbuilt directives for HTML Checkbox**

```
<input type="checkbox" ng-model="isPatientArrived"    ng-true-value="YES"
                                                       ng-false-value="NO"></div>
```

| **Parameter** | **Details** |
|---|---|
| ngModel | Assignable angular expression to data-bind to |
| name*(optional)* | Property name of the form under which the control is published. |
| ngTrueValue*(optional)* | The value to which the expression should be set when selected. |
| ngFalseValue*(optional)* | The value to which the expression should be set when not selected. |
| ngChange*(optional)* | Angular expression to be executed when input changes due to user interaction with the input element. |

* Refer Demo Example - Checkbox_Directive.html

# Inbuilt Directives (4/7)

**Inbuilt directives for HTML Radiobutton**

```
<input type="radio" ng-model="color" ng-value="specialValue"> Green <br/>
  <input type="radio" ng-model="color" value="red">  Red <br/>
```

| Param | Details |
|---|---|
| ngModel | Assignable angular expression to data-bind to. |
| value | The value to which the expression should be set when selected. |
| name*(optional)* | Property name of the form under which the control is published. |
| ngChange*(optional)* | Angular expression to be executed when input changes due to user interaction with the input element. |
| ngValue | Angular expression which sets the value to which the expression should be set when selected. |

# Inbuilt Directives (5/7)

## Inbuilt directives for HTML Textbox

```
<input type="text" name="input" ng-model="patientName"
            ng-pattern="word" required ng-trim="false" ng-minlength=3   ng-maxlength=20>
```

| | |
|---|---|
| Required (optional) | Adds required validation error key if the value is not entered. |
| ngRequired (optional) | Adds required attribute and required validation constraint to the element when ngRequired expression evaluates to true.<br><br>Use ngRequired instead of required when you want to data-bind to the required attribute. |
| ngMinlength (optional) | Sets minlength validation error key if the value is shorter than minlength. |
| ngMaxlength (optional) | Sets maxlength validation error key if the value is longer than maxlength.<br><br>Setting the attribute to a negative or non-numeric value, allows view values of any length. |
| ngChange (optional) | Angular expression to be executed when input changes due to user interaction with the input element. |
| ngTrim (optional) | If set to false Angular will not automatically trim the input.<br><br>This parameter is ignored for input[type=password] controls, which will never trim the input. (default: true) |

## Inbuilt Directives (6/7)

**Inbuilt directives for HTML Number**

```
<input type="number" name="input" ng-model="value"   ng-minlength="0"  ng-maxlength ="99"
        required>
```

| | |
|---|---|
| min(optional) | Sets the min validation error key if the value entered is less than min. |
| max(optional) | Sets the max validation error key if the value entered is greater than max. |
| required(optional) | Sets required validation error key if the value is not entered. |
| ngRequired(optional) | Adds required attribute and required validation constraint to the element<br><br>when the ngRequired expression evaluates to true.<br><br>Use ngRequired instead of required when you want to data-bind to the required attribute. |
| ngMinlength (optional) | Sets minlength validation error key if the value is shorter than minlength. |
| ngMaxlength (optional) | Sets maxlength validation error key if the value is longer than maxlength.<br><br>Setting the attribute to a negative or non-numeric value, allows view values of any length. |

* Refer Demo Example - NumberValidation_Directive.html

## Inbuilt Directives (7/7)

### Inbuilt directives for HTML Email

```
<input type=“email" name="input"   ng-pattern="word" >
```

| | |
|---|---|
| Pattern (optional) | Similar to ngPattern except that the attribute value is the actual string that contains the regular expression body that will be converted to a regular expression as in the ngPattern directive. |
| ngPattern (optional) | Sets pattern validation error key if the ngModel value does not match a RegExp found by evaluating the Angular expression given in the attribute value.<br><br>If the expression evaluates to a RegExp object then this is used directly.<br><br>If the expression is a string then it will be converted to a RegExp after wrapping it in ^ and $characters.<br><br>For instance, "abc" will be converted to new RegExp('^abc$'). |
| ngChange (optional) | Angular expression to be executed when input changes due to user interaction with the input element. |

*Refer Demo Example - EmailValidation_Directive.html

## Inbuilt Directives

- Demo Link on Email Validation
  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/11_EmailValidation_Directive.html

- Demo Link on Checkbox
  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/10_ng_truevalue_Checkbox_Directive.html

- Demo Link on Number Validation
  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/12_NumberValidation_Directive.html

- Demo Link on Radio Button using inbuilt Directive
  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/13_ngValueRadioButton_Directive.html

- Demo Link on TextDirective using ngPattern
  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/14_ngPattern_Text_Directive.html

- Demo on ng-hide and ng-show
  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/15_store2_nghide_ngshow.html

## Agenda

- What Are Directives?
- Matching Directives
- Inbuilt Directives
- **Custom Directives**
- Use of Restrict Function in Directives
- Isolating Scope of the Directive
- Dom Manipulation & Adding Event Listeners

## Custom Directives (1/3)

**How to register custom directives?**

- Directives are registered on modules.
- Module.directive API is used to register directive.

```
var module = angular.module(‘simpleDirective', []);
module.directive (‘patientTagline', function() {
                return {
                        ….
                };
 });
```

- Module.directive takes the directive name followed by a factory function, which in turn returns an object with different options to inform compiler how directive should behave when DOM matches.
- Factory function is invoked when the compiler matches the directive first time.
- Basically used to perform any initialization work.

**Best practice for directive name**

- Prefer to use your own prefix with directive name to avoid collision with some standard feature of HTML
- Do not prefix directive with “ng” to avoid conflict with existing or future directives.

## Custom Directives (2/3)

- When to use custom Directive?
  - Assume if you have a piece of code that display patient information & it is repeated many times in your code & change in one place will lead to change in several places
  - In above case, use a directive to simplify your template.

  * Refer Demo on CD_Basic.html
  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/1_CDBasic.html

- In the above example, we in-lined the value of template option, this will become useless when size of template grows.
- In this case, it's better to use template in external HTML file & load it with the "**templateUrl**" option.

  * Refer Demo on - CD_Template.html, CD_Template2.html
  http://ctgit/silviap/angularjsdemosandlabs/edit/master/Day2/day2_Demos/3_CDTemplate/CD_Template.html

  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/4_CDTemplate2/CD_Template2.html

CitiusTech

## Custom Directives (3/3)

**Template Expanding Directives**

- “templateUrl” can also be written as function
- Angular will invoke“templateUrl” function with two parameters
  - The element that directive was called on
  - Attribute object associated with that element.
- This function will finally return an URL of the HTML template to be loaded.

*Refer Demo on - CD_TemplateUrlAsFunction.html

http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/5_CD_TemplateUrlAsFunction/CD TemplateUrlAsFunction.html

## Agenda

- What Are Directives?
- Matching Directives
- Inbuilt Directives
- Custom Directives
- **Use of Restrict Function in Directives**
- Isolating Scope of the Directive
- Dom Manipulation & Adding Event Listeners

## Use of Restrict Function in Directives

- By default , custom directives can be used as attributes and elements only.
- To allow usage of custom directive to attribute/element only  “restrict” option  can be used.
- To trigger directives by class name , the “restrict” options are used.

* Refer Demo on  - CD_UseOfRestrict.html

  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/2_CD_UseOfRestrict/CD_UseOfRestrict.html

Restrict Options

- 'A' - only matches attribute name
- 'E' - only matches element name
- 'C' - only matches class name

Above restriction can all be combined as needed.

- 'AEC' - matches either attribute or element or class name

## Agenda

- What Are Directives?
- Matching Directives
- Inbuilt Directives
- Custom Directives
- Use of Restrict Function in Directives
- **Isolating Scope of the Directive**
- Dom Manipulation & Adding Event Listeners

## Isolating Scope of the Directive

- If we want to use same custom directive but to show different information based on attribute value of directive , we need to use isolated scope.
- For e.g.: Same 'patient-Tagline' custom directive is to be used to show different records of patient1 and patient2

```
<div ng-controller="PatientController">
                        <patient-Tagline   info="patient1"></patient-Tagline>
                        <hr/>
                        <patient-Tagline    info="patient2"></patient-Tagline>

</div>
```

* Refer Demo on - CD_IsolateScope.html
http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/6_CD_IsolateScope/CD_IsolateScope.html

- To achieve above we need to map controller's scope to directive's inner scope
- Directive's "scope" option is used to achieve this.
- In Demo – 'CD-IsolateScope.html' , First <patient-Tagline> directive element binds the "info" attribute to "patient1", which is exposed in scope of controller.

## Agenda

- What Are Directives?
- Matching Directives
- Inbuilt Directives
- Custom Directives
- Use of Restrict Function in Directives
- Isolating Scope of the Directive
- **Dom Manipulation & Adding Event Listeners**

## Create Directives that Manipulate the DOM

- To modify the DOM via  custom directive, “link” function is used
- It takes a function with the following signature

```
function link(scope, element, attrs){
        …
}
```

- **scope** is an Angular scope object.
- **element** is the element that this directive matches.
- **attrs** is a hash object with key-value pairs of normalized attribute names and their corresponding attribute values.

*Refer Demo on - CD_ManipulateDOM.html

[http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/8_CD_ManipulateDOM/CD_ManipulateDOM.html](http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/8_CD_ManipulateDOM/CD_ManipulateDOM.html)

- We don’t directly refer to element or attribute using it’s name  inside link function, this avoids tight-coupling with Html DOM

## Create Directives that wraps other Elements

- In Isolating scope example, we can pass the model to the directive.
- Sometimes it requires to pass the entire template rather than string or object & that should wrap any arbitrary content.
- To achieve the above functionality “transclude” option is used.

*Refer Demo on - CD_WrapElements.html

http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/9_CD_WrapElements_Transclude/CD_WrapElements.html

- “transclude” is used to make the content of the directive with this option have access to the scope outside of the directive rather than inside.
- The “transclude” option changes the way scopes are nested.
- Contents of a transcluded directive refers to the outside scope of the directive, rather than whatever scope is on the inside.
- In doing so, it gives the contents access to the outside scope.

## Create Directives to Add Event Listeners

- Angular provides capabilities to create directives that react to events on its elements

  i.e. mouseOver, mouseOut, mouseUp, mouseDown etc…

*Refer  Demo on - CD_EventListener.html

  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/9_CD_WrapElements_Transclude/CD_WrapElements.html

## Custom Directive - Finding your way (1/2)

Technical Questions:

1. Explain the scenarios where "transclude" option of custom directives can be used?

2. Explain How Angularjs custom directive 'link' function allows to do DOM manipulation without getting tightly coupled with Html DOM

**As you start to work with AngularJS CustomDirectives, you will frequently encounter technical issues which are not covered by this training.**

**How will you resolve these technical issues on AngularJS Custom Directives?**

# Custom Directive -Finding your way (2/2)

| Resources | Remarks |
|---|---|
| | |
| http://tutorials.jenkov.com/angularjs/custom-directives.html | Good blog post on Angularjs custom directives |
| https://docs.angularjs.org/guide/directive | AngularJS official doc on Directives |
| https://github.com/jedrichards/angularjs-handbook#directives | Good handbook on AngularJS that includes directives. |
| | |
| CurioCT - https://interct/SitePages/CurioCT.aspx | |
| CTCourses – Custom Directives | In addition to updated course material, CTCourse contains reference sites (Library) and list of project teams with expertise on Directives |
| | |

## CurioCT - CitiusTech's Technology Q & A Forum

- In case of any questions please log on to https://interct/SitePages/CurioCT.aspx

# Thank You