AngularJS - Introduction to Customized Directives

By

Sarada Pachalla
Lead UI Developer @

GE
What are Directives?

• Functions called on a DOM element to augment its functionality
• Teach browsers new HTML tags
• Two types of Directives –
  – Built-in Directives
  – Custom Directives
Built-in Directives

• ng-click
• ng-show/ng-hide
• ng-model
• ng-repeat
• ng-controller
• ng-if
• Many more...
Custom Directives

• What are Custom Directives?
• When to use Custom Directives?
• Naming conventions
• Getting Started
Custom Directives

• Functions written by users to create new HTML tags, attributes or class
• Secret sauce of Angular
• Expressiveness is the real power of writing custom directives
When do we use ...

• Almost any time you need to interface with the DOM.
• If we want a re-usable HTML component or for complex UI expressions
  – `<my-widget>...</my-widget>`
  – `<div ng-include=""..."> ...
• If you want reusable HTML behavior like events handlers
  – `<div ng-click="..."> ...
• If you want to wrap a jQuery plugin
  – `<div ui-date>...

Naming Conventions

• Use a unique prefix
  – Avoids clashing with others
  – Easier for readers to identify
• Don’t use “ng-“ for your custom prefix
• Common convention: two letters
  – The AngularUi project uses “ui-“
Getting Started

1. Create a module for your directives:
   angular.module(‘MyDirectives’, []);
2. Load the module in your app:
   angular.module(‘MyApp’, [‘MyDirectives’]);
3. Register your directive:
   angular.module(‘MyDirectives’)
   .directive(‘myDtv1’, function(){
     //TODO ... write code for directive here
   });
4. Use your directive in your HTML:
   <div my-dtv1>...</div>
A basic pseudo-code template for creating any directive:

```javascript
angular.module(...);
directive('namespaceDirectiveName', function factory(injectables) {
  var directiveDefinitionObject = {
    restrict: string,
    priority: number,
    template: string,
    templateUrl: string,
    replace: bool,
    transclude: bool,
    scope: bool or object,
    controller: function controllerConstructor($scope,

    $element, $attrs, $transclude), require: string,
    link: function postLink(scope, iElement, iAttrs) { ... }, compile: function compile(tElement, tAttrs, transclude) {
    return {
      pre: function preLink(scope, iElement, iAttrs, controller) { ... }, post: function postLink(scope, iElement, iAttrs, controller) { ... }
    }
  }
  return directiveDefinitionObject; });
```

Some of the options are mutually exclusive, most of them are optional, and all of them have details that are worth explaining.
Thanks You!

@saradapachalla