AngularJS Directives

Objectives

- What are directives?
- The Directive Definition Object
- Link & Compile functions
- Using templates
- Transclusion
- Isolated scope
- Change notification

What are directives?

- Directives are markup extensions to trigger specific behavior
  - Processed by the Angular HTML compiler $compile
- Created using module.directive()
  - Defined through a Directive Definition Object
- Can be used as:
  - Attributes
  - Elements
  - CSS Classes
  - Comments
- Turn your markup into a DSL using custom directives

Verbose Markup

- **What does this do?**

```
<div ng-controller="peopleCtrl as ctrl" class="container">
    <form name="form" class="form-horizontal" role="form">
        <div class="form-group">
            <label class="col-lg-2 control-label">Firstname</label>
            <div class="col-lg-8">
                <input type="text" class="form-control" ng-model="person.firstName" />
            </div>
        </div>
        <div class="form-group">
            <label class="col-lg-2 control-label">Lastname</label>
            <div class="col-lg-8">
                <input type="text" class="form-control" ng-model="person.lastName" />
            </div>
        </div>

        <div class="form-group">
            <div class="col-lg-offset-2 col-lg-8">
                <button ng-click="ctrl.save()" class="btn btn-primary">Save</button>
            </div>
        </div>
    </form>
</div>
```

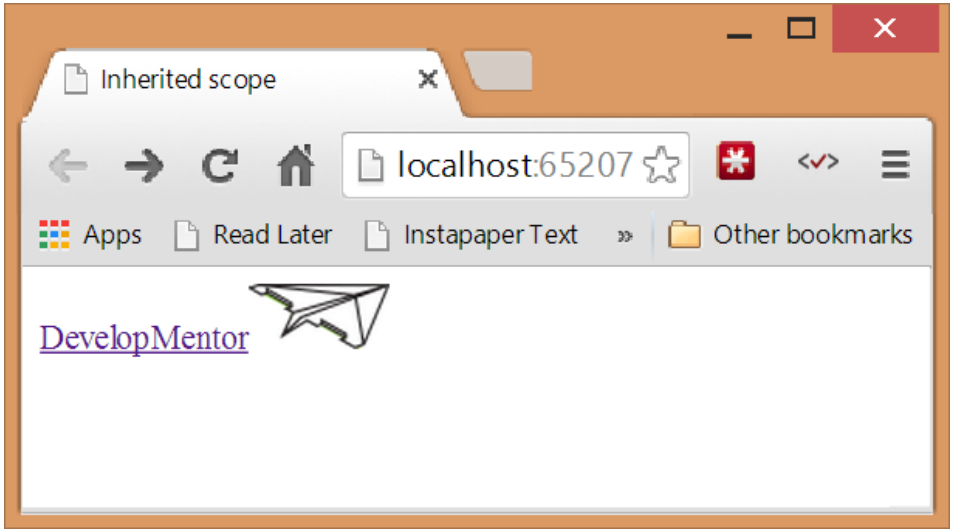
Angular as a Domain Specific Language

- **What does this do?**

```
<div ng-controller="peopleCtrl as ctrl"
     class="container">

    <my-person-editor model="person"
                      on-save="ctrl.save()" />
</div>
```

Example: Creating a reusable logo

```
<body ng-app="myApp">

    <dm-logo />

    <script src="/Scripts/angular.js"></script>
    <script src="App.js"></script>
</body>
```



Example: Creating a reusable logo

- Only usable as an element
- Replaces the element with the template

```
var app = angular.module('myApp', []);

app.directive('dmLogo', function () {
    var ddo = {
        restrict: 'E',
        replace: true,
        template: '<a href="http://www.develop.com/">' +
            '<span>DevelopMentor</span>' +
            '<img src="/images/v3/logos/dm-logo-airplane.png">' +
            '</a>'
    };

    return ddo;
});
```

Example: Setting focus to an element

```
<body ng-app="myApp">

    <input type="text" value="" dm-set-focus="“true" />

    <script src="/Scripts/angular.js"></script>
    <script src="App.js"></script>
</body>
```

Example: Setting focus to an element

- Only usable as an attribute
- Link function does the actual work

```
var app = angular.module('myApp', []);

app.directive('dmSetFocus', function () {
    var ddo = {
        restrict: 'A',
        link: function (scope, element) {
            element[0].focus();
        }
    };

    return ddo;
});
```

The Directive Definition Object

- Defines a directive for the compiler
  - scope
  - require
  - restrict
  - link/compile
  - priority
  - controller
  - template/templateUrl
  - replace
  - transclude

Restricting a directive

- The restrict property determines how a directive can be used
  - Multiple options are possible
- Defaults to Attribute if omitted
- Options
  - E : Element
    <my-menu title=Products></my-menu>
  - A : Attribute
    </my-menu>
  - C : CSS Class

  - M : HTML Comment
    <!-- directive: my-menu Products -->

Link & Compile functions

- Allow full access to the DOM objects
- Normally only the link function is needed
  - Compile function works with templates

The link function

- The link function is used to set register DOM event handlers
  - Has access to the current scope
- Executed one for each element instance
  - After templates have been cloned
- The element passed is a jqLite element
  - A stripped down version of jQuery
  - If jQuery is included it will be a full jQuery object

The link function

```
var app = angular.module('myApp', []);

app.directive('myDirective', function () {
    return {
        link: function (scope, iElement, iAttrs, ctrl, tran) {
            iElement.on('click', function () {
                console.log('Clicked on', iElement);
            });
        }
    };
});
```

The compile function

- The compile function transforms the template DOM
  - Not that commonly used
- Cannot be combined with a link function in the DDO
  - Instead return the link function from the compile function

Combining link & compile functions

```
<body ng-app="myApp">
    <div my-directive>
        Lorem ipsum dolor sit amet.
    </div>

    <ul>
        <li ng-repeat="val in ['One','Two','Three']"
            my-directive>{{val}}</li>
    </ul>
</body>
```

```
app.directive('myDirective', function () {
    return {
        compile: function compile(tElement, tAttrs) {
            console.log('compiling', tElement);

            return function link(scope, iElement, iAttrs, ctrl, tran) {
                console.log('linking', scope, iElement);
            };
        }
    };
});
```

Combining link & compile functions

Inherited scope

localhost:65207/Scopes/Inherited.html

Apps | Read Later | Instapaper Text | Office 365 | eVision Industry Soft... | Online Time Trackin... | Dive Into HTML5 | Other bookmarks

Lorem ipsum dolor sit amet, consectetur adipiscing elit fusce vel sapien elit in malesuada semper mi, id sollicitudin urna fermentum ut fusce varius nisl ac ipsum gravida vel pretium tellus.

- One
- Two
- Three

Elements | Resources | Network | Sources | Timeline | Profiles | Audits | Console | AngularJS

```
compiling ▶[div, ready: function, toString: function, eq: function, push: function, sort: function…]   App.js:7
compiling ▶[li, ready: function, toString: function, eq: function, push: function, sort: function…]   App.js:7
linking ▶Scope {$id: "002", $$childTail: null, $$childHead: null, $$prevSibling: null, $$nextSibling: null…}
▶[div, ready: function, toString: function, eq: function, push: function, sort: function…]   App.js:10
linking ▶Child {$id: "003", this: Child, $$listeners: Object, $parent: Scope, $$childTail: null…}
▶[li.ng-scope ng-binding, ready: function, toString: function, eq: function, push: function, sort: function…]   App.js:10
linking ▶Child {$id: "004", this: Child, $$listeners: Object, $parent: Scope, $$childTail: null…}
▶[li.ng-scope ng-binding, ready: function, toString: function, eq: function, push: function, sort: function…]   App.js:10
linking ▶Child {$id: "005", this: Child, $$listeners: Object, $parent: Scope, $$childTail: null…}
▶[li.ng-scope ng-binding, ready: function, toString: function, eq: function, push: function, sort: function…]   App.js:10
>
```

<top frame> | <page context> | All | Errors | Warnings | Logs | Debug

Using templates

- Templates can be used to insert markup
  - Replaced the inner content unless replace is true
- The original inner content is replaced
  - Unless transclusion is specified
- Templates can be inline or loaded through a templateUrl
  - Inline is only practical for small templates
- A templateUrl is loaded when first needed and cached
  - Can be preloaded as script bock type text/ng-template
  - The id should be the same as the Url

Transclusion

- Inserts the original content into the template
  - Otherwise this is lost
- Add transclude: true to the Directive Definition Object
  - Insert the ng-transclude directive into the template

```
<div my-transcluded-directive>
    <span>Original content</span>
</div>
```

```
app.directive('myTranscludedDirective', function () {
    return {
        template: '<div>From template</div>' +
                  '<div ng-transclude></div>' +
                  '<div>More template</div>',
        transclude: true
    };
});
```

Isolating the Scope

- By default a directive shares it's parents scope
  - Often not a good solution
- A directive can have it's own scope
  - Inherited scope
  - Isolated scope
- A DOM element can be associated with a single scope
  - The most restrictive is applied

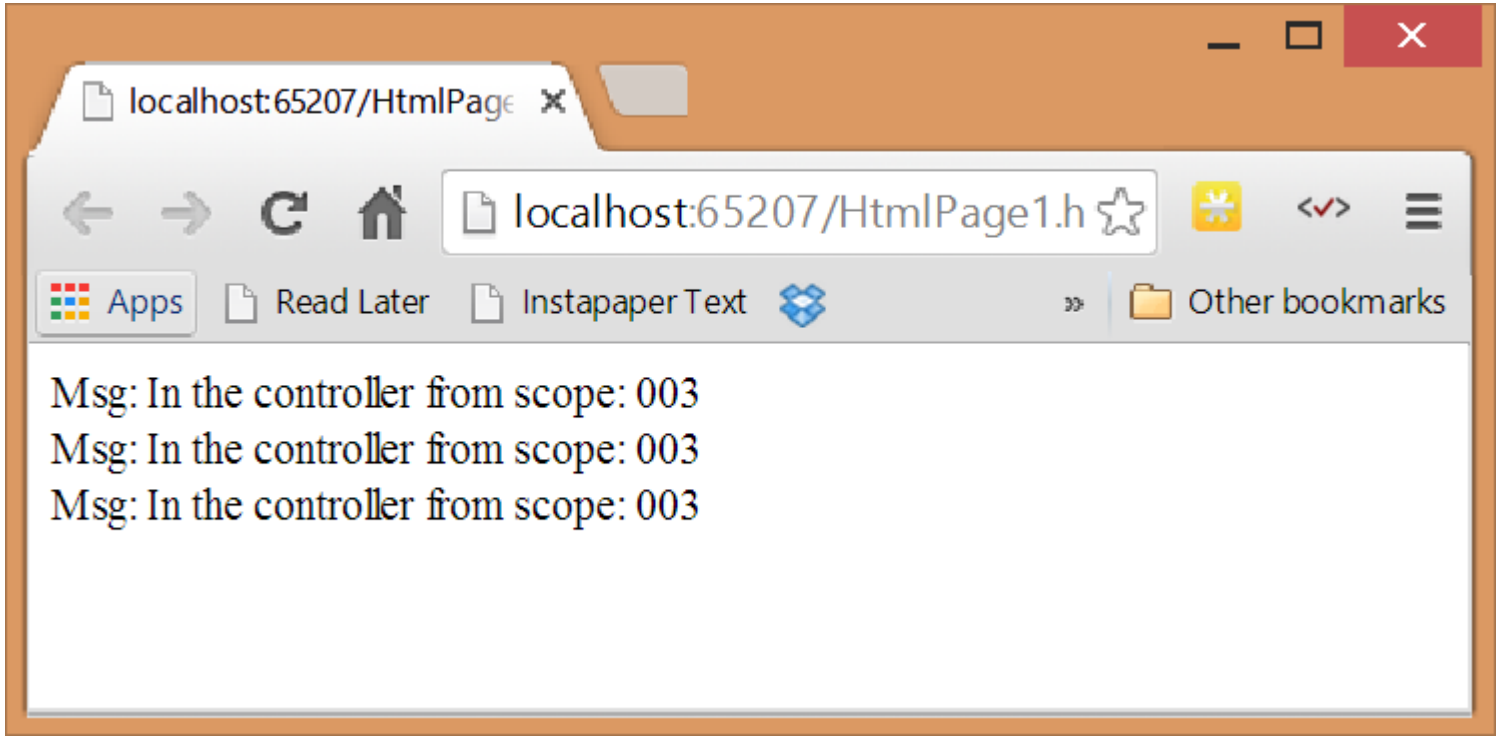
No scope

- Uses its parent scope

```
<div ng-controller="myCtrl">
    <div>Msg: {{msg}} from scope: {{$id}}</div>
    <div my-unscoped-directive></div>
    <div my-unscoped-directive></div>
</div>
```

```
app.controller('myCtrl', function ($scope) {
    $scope.msg = “In the controller";
});

app.directive('myUnscopedDirective', function () {
    return {
        template: '<div>Msg: {{msg}} from scope: {{$id}}</div>',
    };
});
```

No scope



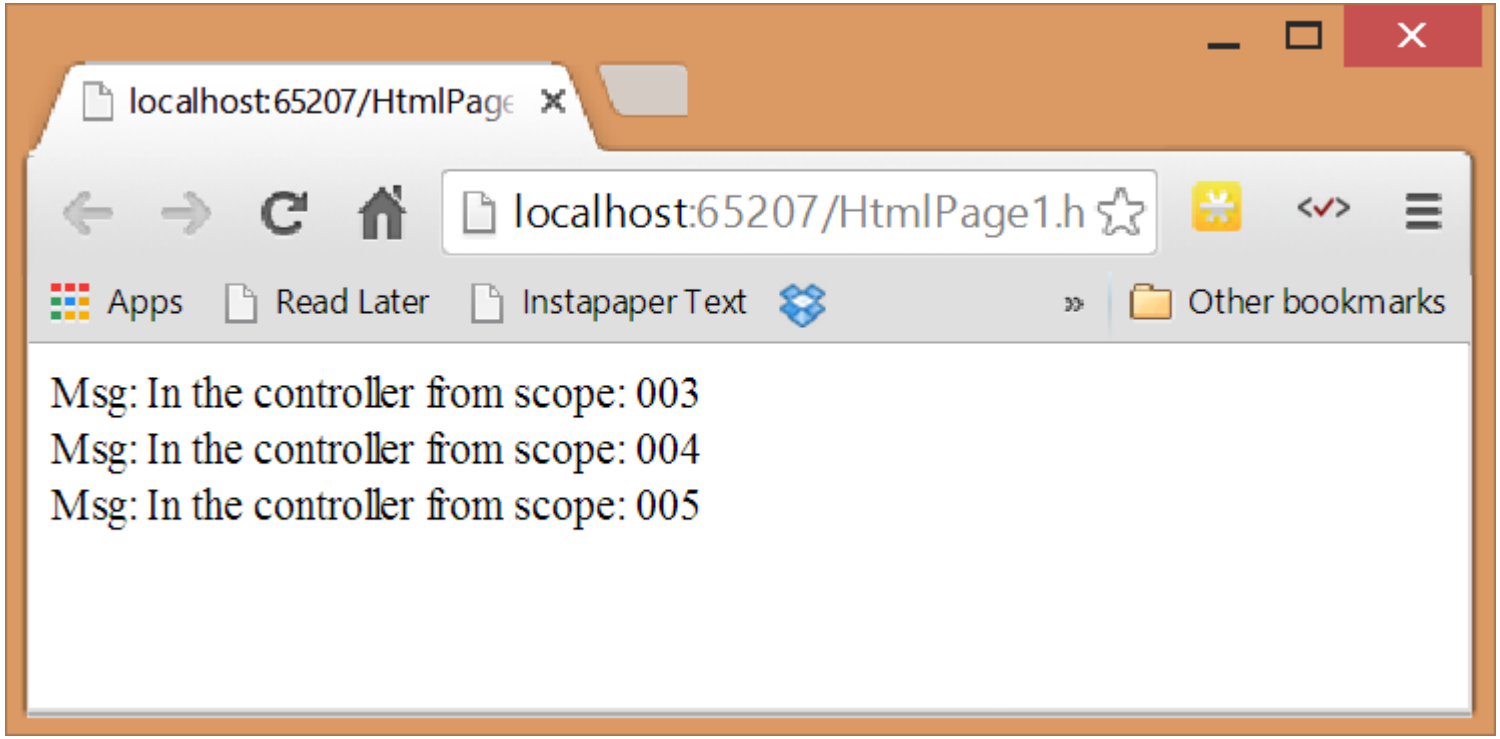
Inherited scope

- Creates a new scope
  - Prototypically linked to its parent scope

```
<div ng-controller="myCtrl">
    <div>Msg: {{msg}} from scope: {{$id}}</div>
    <div my-linked-scoped-directive></div>
    <div my-linked-scoped-directive></div>
</div>
```

```
app.directive('myLinkedScopedDirective', function () {
    return {
        template: '<div>Msg: {{msg}} from scope: {{$id}}</div>',
        scope: true
    };
});
```

Inherited scope



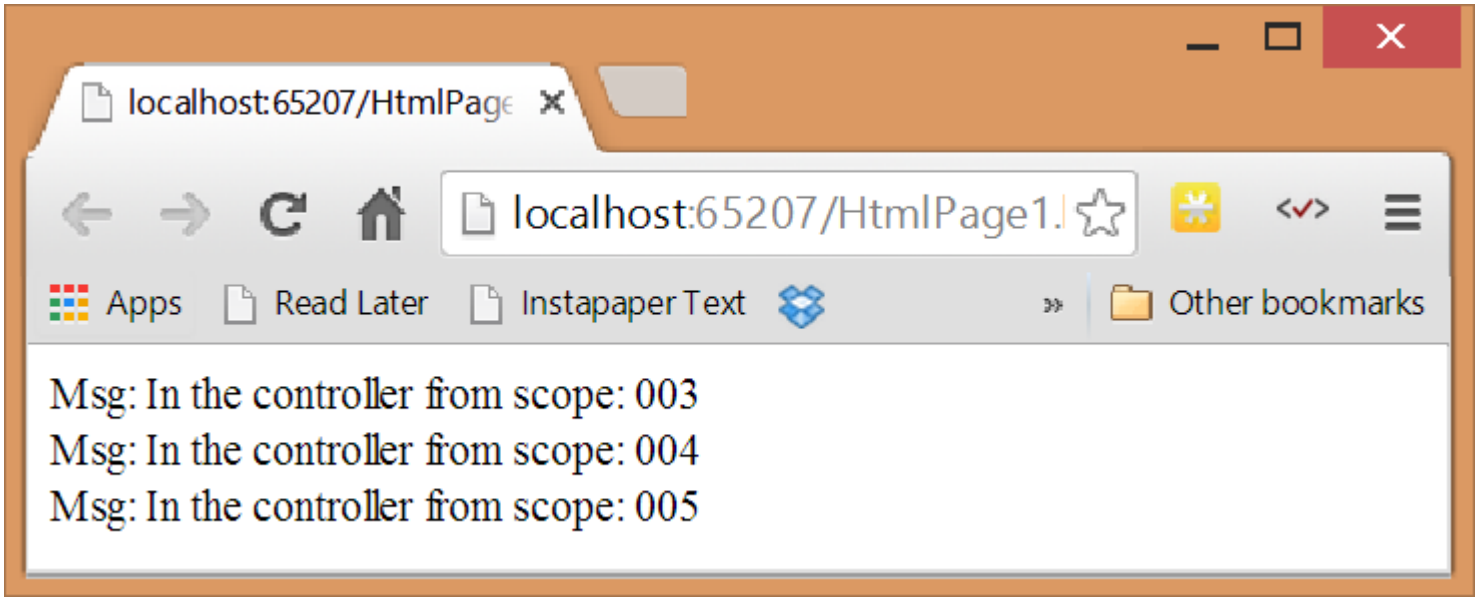
Isolated scope

- Creates a new scope
  - Not linked to its parent scope

```
<div ng-controller="myCtrl">
    <div>Msg: {{msg}} from scope: {{$id}}</div>
    <div my-isolated-scoped-directive the-msg="msg"></div>
    <div my-isolated-scoped-directive the-msg="msg"></div>
</div>
```

```
app.directive('myIsolatedScopedDirective', function () {
    return {
        template: '<div>Msg: {{theMsg}} from scope: {{$id}}</div>',
        scope: { theMsg: '=' }
    };
});
```

Isolated scope



Isolated scope

- The scope has no prototypical link to the parent scope
  - Properties are not just available
  - Can be reached explicitly via $parent
- Populate the scope with:
  - Parent scope items using =
  - Attribute strings using @
  - Function expressions from parent scope using &

Require

- Requires another directive to the present
  - Can be optional or located on the parent element
  - Possible to specify an array if needed
- Use $setViewValue() to update the underlying model
  - Updates ngForm values like $pristine and $dirty

```
return {
        require: "ngModel",
        link: function (scope, element, attrs, ctrl) {
                ctrl.$setViewValue("New Value");
        }
};
```

Change notification

- The $watch function on the scope will notify of changes
  - Use either a watch function or expression string
- The $watch function will be called often!
  - Be careful with performance
- Specify true as the third parameter for a deep watch
  - Also watched nested objects

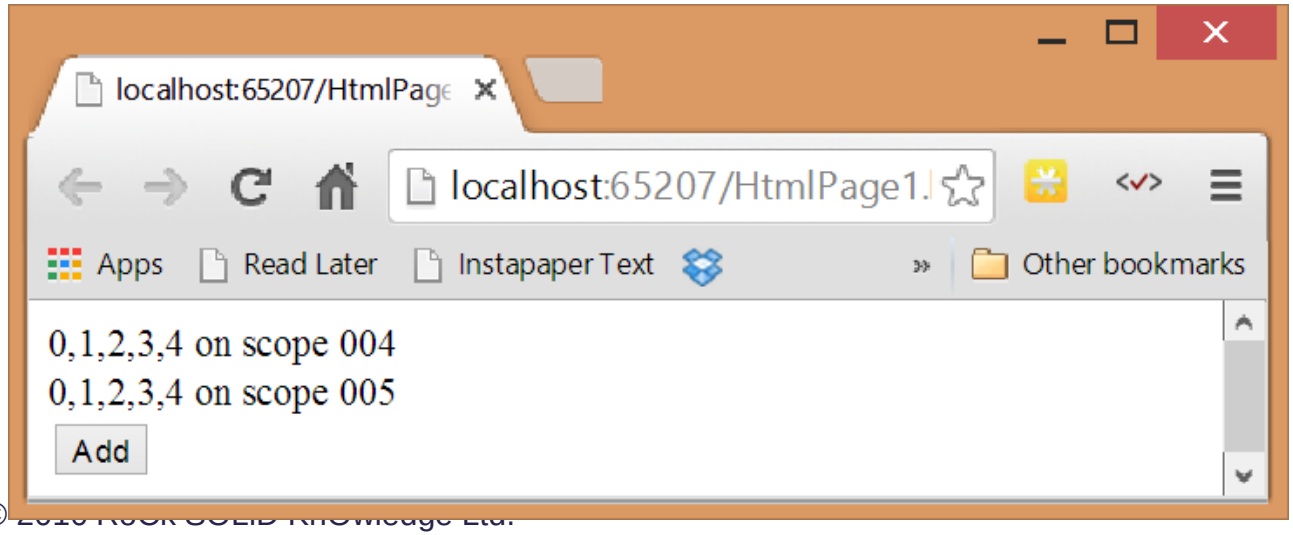
Change notification

```
<div ng-controller="myCtrl">
    <div my-watching-directive values="values"></div>
    <div my-watching-directive values="values"></div>
    <button ng-click="addNumber()">Add</button>
</div>
```

```
app.controller('myCtrl', function ($scope) {
    $scope.values = [0, 1];

    $scope.addNumber = function () {
        $scope.values.push($scope.values.length);
    };
});
```

Change notification

```
app.directive('myWatchingDirective', function () {
    return {
        link: function (scope, element) {
            scope.$watch('values', function (newVal) {
                element.text(newVal.toString() +
                    " on scope " + scope.$id);
            }, true);
        },
        scope: { values: '=' }
    };
});
```



Cleaning up

- Sometimes you need to do cleanup work in a directive
  - Unbind event handlers
- Add an event handler for the $destroy event
  - Either on the element or the scope

- element.on('$destroy', …)
- scope.$on('$destroy', …)

Summary

- Directives are markup extensions to trigger specific behavior
  - Defined using the Directive Definition Object
- Use the link function to add DOM event handlers
- Using templates to insert new markup
  - Transclude markup into templates where needed
- Isolate the directives scope as needed
  - Only include the data you really need
- Use change notification for updates
  - Do not forget to clean up after yourself