Custom Directives

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Kinds of Directives

- Widgets
  - ex. playing card, dialog, date picker
- Event handling
  - primarily DOM events
  - ex. **`ng-click`** and **`digits-only`** (to limit accepted keystrokes in an input)
- Functionality
  - ex. **`ng-include`** and **`ng-show`**
- Splitting large pages into smaller parts

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Isolate Scope

- A term that comes up frequently when describing directives
- A scope that does not inherit from any other scope ... it's isolated
- Directives that use isolate scope are
  easier to reuse in different contexts
  because they do not assume that
  any particular data is available in an ancestor scope
- More detail on this later

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Transclusion

- From Wikipedia ...
- "In computer science, transclusion is
  the inclusion of a document or part of a document
  into another document by reference."
- "The term was coined by hypertext pioneer Ted Nelson in 1963."
- We'll see how directives can use this later

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Names

- Choose names that are unlikely to conflict with names of
  - HTML elements and attributes, including those added in HTML5
  - built-in AngularJS directives
  - directives defined in third party libraries you may use
- When multiple directive definitions use the same name, all will be processed when the directive name is used
  - differs from controllers, services and filters where the last definition wins and previous definitions are lost
  - avoid this!
- Consider including a name prefix
  - perhaps related to your application, library or company
- Name in JavaScript definition must be camel-case - ex. **`fooBarBaz`**
- Name in HTML are snake-case - ex. **`foo-bar-baz`**
  - actually can use any combination of colon, hyphen and underscore delimiters, but hyphen is typical

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Other Kinds of Conflicts

- When multiple directives are applied to the same DOM element, they always share the same scope, even if they are defined to use an isolate scope
  - if one asks for an isolate scope, all of them will share an isolate scope
- It is a conflict if multiple directives on the same element use any of these options
  - **`template`** or **`templateUrl`**
  - **`transclude`**

these options are described later

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Directive Definitions

- Directives are defined using the `directive` `Module` method

```
module.directive('name', function () {
  return directive-definition-object;
});
```

- Name must be camel-cased,
  but refer to in HTML with snake-case name
  - ex. `fooBarBaz` -> `foo-bar-baz`
- The directive definition object contains
  a subset of the properties described on the next slides
  - their meaning will be more clear after seeing examples of each
  - listed in approximate order of use,
    from most to least often used

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Directive Definition Properties ...

- **`restrict`**
  - string that contains a subset of the following characters that indicate how the directive can be used:
    `E` = element, **`A`** = attribute, **`C`** = class, **`M`** = comment
  - defaults to attribute-only

see slide 3 in "Provided Directives" section for usage syntax

- **`template`**
  - string of HTML (inline template)
  - suitable for small templates (a few lines)
- **`templateUrl`**
  - string path to an HTML file (external template)
  - better for large templates
- **`replace`**
  - if true, element on which directive appears, and its content, is replaced by directive template and attributes are copied
  - if false, content of element on which directive appears is replaced by directive template (the default)

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Directive Definition Props ...

- **`scope`**
  - boolean or an object that defines an isolate scope to be used by this directive
  - more detail later
- **`link`**
  - function that takes **`scope`**, **`element`**, **`attrs`** arguments
    - positional parameters, not injectable
  - the scope provided is controlled by **`scope`** property above
  - can add data and functions to scope just like in a controller
  - can perform DOM manipulations on **`element`** and its descendants
  - **`attrs`** provides access to attributes, including those specified in an isolate scope
  - called once for each directive instance
  - will see many examples later

Three ways to add properties to the scope of a directive:
1) isolate scope from attributes
2) link function - most common
3) controller function - typically only used when another directive wants to share the controller using **`require`** or when this directive doesn't need access to the **`element`** and **`attrs`** parameters of the **`link`** function

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

… Directive Definition Props …

- **`transclude`**
  - enables including content of element where directive appears inside directive template
  - provided directives that use this include **`ng-if`**, **`ng-switch`** and **`ng-repeat`**
  - see dialog example later
- **`controller`**
  - injectable function (typically inject **`$scope`**) or the string name of a controller defined elsewhere
  - manages the scope used by template, assigning data and functions to scope properties that are used in binding expressions and directives in the template
  - if isolate scope is not being used, can also access data and functions in ancestor scopes
  - setting to a string name allows multiple directives to share the same controller
  - don't perform DOM manipulation here
- **`priority`** not commonly used
  - integer that controls order in which directives on same element will be processed
  - default is zero; highest numbers go first; use negative numbers to process after default

in most cases, **`link`** can be used instead of **`controller`**; see http://jasonmore.net/angular-js-directives-difference-controller-link/

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Directive Definition Properties

- **`require`** not commonly used
  - string name of another directive, or array of them, that must be present on the same element or ancestor elements (with "`^`" prefix); more detail later
  - allows multiple directives to communicate through a shared controller
- **`compile`** not commonly used
  - a function that is passed the template element and its attributes
    - typically these parameters are named **`tElement`** and **`tAttrs`**
  - only purpose is to modify template
    - important when **`ng-repeat`** directive or similar iterating directives might appear on same element
  - can return
    - a link function that takes scope, instance element (**`iElement`**), instance attributes (**`iAttrs`**), controller and transclude function (**`transcludeFn`**)
    - an object with properties **`pre`** and **`post`** that are pre-link and post-link functions
  - if present, the **`link`** function is ignored

  see examples/directives/pre-post-link

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Common `restrict` Values

- `'E'` means `<some-long-name>optional-content</some-long-name>`
- `'A'` means `<div some-long-name>optional-content</div>`
- Class and comment directives are rarely used
  - very little justification for using them

- To use element directives in IE8, a DOM element must be created with the directive name

```
document.createElement('some-long-name');
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Basic Example

index.html

```
<!DOCTYPE html>
<html ng-app="Demo">
  <head>
    <script src=".../angular.min.js"></script>
    <script src="demo.js"></script>
  </head>
  <body>
    <div>You should see "static content" four times below.</div>
    <div rmv-a-c-e></div>
    <div class="rmv-a-c-e"></div>
    <rmv-a-c-e></rmv-a-c-e>
    <!-- directive: rmv-m -->
  </body>
</html>
```

You should see "static content" four times below.
static content
static content
static content
static content

demo.js

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

app.directive('rmvACE', function () {
  return {
    restrict: 'ACE', // A=attribute, C=class, E=element
    template: 'static content'
  };
});

app.directive('rmvM', function () {
  return {
    replace: true,
    restrict: 'M', // M=comment
    template: '<div>static content</div>'
  };
});
```

comment directives must set **replace** to **true** and the template must have a root element

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Breaking Up Large Pages

- Use a separate controller or custom directive for each page section instead of using one controller for the entire page
- To use separate controllers
  - replace each page section with an element like this:
  - `<div ng-controller="ctrl-name" ng-include="'html-template-file-path'"></div>`
  - note single-quotes around file path
  - alternatively, **`ng-controller`** directive can be specified on root element of template
    - first approach is better if template might be used in multiple pages with different controllers
- To use custom directives
  - replace each page section with an element like this:
  - `<div directive-name></div>`
  - these directives can be defined in separate source files
  - see next slide

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Splitting Page With Directives

- Each section of a page can be specified and managed by a custom directive

some.html

```
<div>
  <div part1></div>
  <div part2></div>
  <div part3></div>
</div>
```

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

module.directive('part1', function () {
  return {
    restrict: 'AE',
    replace: true,
    controller: 'Part1Ctrl',
    templateUrl: 'feature/some-name/part1.html'
  };
});

module.controller('Part1Ctrl', function ($scope) {
  // Add data and functions to scope that part1.html needs.
});

// Add directives and controllers for other parts.
// Components for each part can be defined in separate source files.
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Passing Data ...

Hello, Moe Howard!

Hello, Larry Fine!

Hello, Curly Howard!

Hello, Shemp Howard!

demo.css

```
.hello {
  background-color: LightSalmon;
  border: solid red 3px;
  border-radius: 5px;
  font-family: sans-serif;
  margin: 5px;
  padding: 5px;
  width: 165px;
}
```

index.html

```
<!DOCTYPE html>
<html ng-app="Demo">
  <head>
    <script src=".../angular.min.js"></script>
    <link rel="stylesheet" href="demo.css">
    <script src="demo.js"></script>
  </head>
  <body>
    <div class="hello" hello-world1 first="Moe" last="Howard"></div>
    <div class="hello" hello-world2 first="Larry" last="Fine"></div>
    <div class="hello" hello-world3 first="Curly" last="Howard"></div>
    <div class="hello" hello-world4 first="Shemp" last="Howard"></div>
  </body>
</html>
```

showing four approaches to writing this directive, but the same could be used for all of these

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Passing Data ...

reusing existing scope

demo.js

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

app.directive('helloWorld1', function () {
  return {
    template: 'Hello, {{first}} {{last}}!',
    link: function (scope, element, attrs) {
      scope.first = attrs.first;
      scope.last = attrs.last;
    }
  };
});
```

explicitly getting attribute values as strings

using isolate scope

demo.js

```
app.directive('helloWorld2', function () {
  return {
    scope: {
      first: '@',
      last: '@'
    },
    template: 'Hello, {{first}} {{last}}!',
  };
});
```

getting attribute values as strings via isolate scope

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Passing Data

using isolate scope

demo.js

```
app.directive('helloWorld3', function () {
  return {
    scope: {
      first: '@',
      last: '@'
    },
    template: 'Hello, {{first}} {{last}}!',
    link: function (scope, element, attrs) {
      if (!attrs.first) throw new Error('first attribute is required');
      if (!attrs.last) throw new Error('last attribute is required');
    }
  };
});
```

making attributes required

demo.js

```
app.directive('helloWorld4', function () {
  return {
    // When replace is true, the template must have a root element.
    // Attributes on the element where the directive is applied
    // are copied to that root element.
    replace: true,
    scope: {
      first: '@',
      last: '@'
    },
    // The root element is "p".  This will replace the "div".
    template: '<p>Hello, {{first}} {{last}}!</p>'
  };
});
```

replacing element on which the directive appears

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Directive Scope

- Shared scope - `scope: false` (default)
  - directive doesn't get its own scope
  - uses same scope as containing element
  - if directive sets new scope properties, parent scope will see them
- Inherited scope - `scope: true`
  - directive gets its own scope that inherits from scope of parent element
  - can access all properties in ancestor scopes, unless a property with the same name is defined in this scope (hides ancestor scope properties)
  - if directive sets new scope properties, parent scope will not see them
- Isolate scope - `scope: {...}` can be an empty object
  - directive gets its own scope which is not in scope hierarchy of containing element scope; it is isolated
  - if directive sets new scope properties, parent scope will not see them
  - makes directive reusable across pages of same app and multiple apps
  - more detail on next slide

Both **shared** and **inherited** scope are discouraged for directives that may be reused in different scopes because they can increase coupling by assuming certain scope properties will be present. It's better to use **isolate** scope which limits scope access to specified properties.

avoid by setting a property in an object on parent scope;
`scope.parent-prop.name = value;`
or explicitly setting on parent scope;
`scope.$parent.name = value;`

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Isolate Scope ...

- When **`scope`** property of directive definition object is an object, its properties are names of properties to be added to its isolate scope
- Values can be supplied from attributes of element on which directive appears
  - attribute name must be snake-case version of camel-case scope property name
  - ex. to share property **`fooBarBaz`**, specify attribute as **`foo-bar-baz="`*`value`*`"`**
- Code in directives with isolate scope can still access properties in ancestor scopes, including the root scope, but doing this is discouraged
  - **`scope.$parent.`*`name`***
  - **`scope.$root.`*`name`***

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Isolate Scope

- Values associated with scope property names can be
  - `'@'` - when value is a string
    - corresponding attribute value can contain binding expressions that are evaluated in containing scope to build the string
  - `'='` - to use same object as in containing scope, not a copy
    - by reference
    - ex. `foo: '='`
  - `'&'` - when value is an expression to be evaluated in containing scope
    - typically the expression calls a function on the containing scope
    - often used in conjunction with `ng-repeat` to call a method on current iteration value
  - any literal value - when value to assign is not specified in an attribute
    - boolean, number, string, array or object
- To use a different property name than the attribute name, follow symbol with camel-cased attribute name
  - ***`scopePropertyName: '=camelAttrName'`***

so not exactly "isolated"

no error is thrown if the element does not have a matching attribute

camelcase version of attribute name

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Playing Cards ...

reusable widget that uses isolate scope

index.html

```
<!DOCTYPE html>
<html ng-app="CardGame">
  <head>
    <link rel="stylesheet" href="cards.css"></script>
    <link rel="stylesheet" href="playing-card.css"></script>
    <script src=".../angular.min.js"></script>
    <script src=".../angular-sanitize.min.js"></script>
    <script src="playing-card.js"></script>
    <script src="cards.js"></script>
  </head>
  <body ng-controller="CardGameCtrl">
    <h3>Your Hand</h3>
    <div ng-repeat="card in hand" playing-card="card"></div>
    <div>
      <button ng-click="deal()" ng-show="hand.length === 5">Deal</button>
      <button ng-click="newDeck()">New Deck</button>
    </div>
  </body>
</html>
```

`card` is an object with `rank` and `suit` properties

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

… Playing Cards …

cards.css

```
body {
  font-family: sans-serif;
}

button {
  background-color: LightBlue;
  border-radius: 5px;
  font-weight: bold;
  padding: 10px;
}
```

playing-card.css

```
.playing-card {
  border: solid black 1px;
  border-radius: 5px;
  display: inline-block;
  font-size: 24pt;
  margin: 5px;
  padding: 5px;
  text-align: center;
  width: 50px;
}
```

cards.js

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

app.controller('CardGameCtrl', function ($scope, playingCardSvc) {
  $scope.deal = function () {
    $scope.hand = playingCardSvc.dealHand(5);
  };

  $scope.newDeck = function () {
    playingCardSvc.newDeck();
    $scope.deal();
  };

  $scope.deal();
});
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Playing Cards ...

playing-card.js

```
(function () {
  'use strict';

  // ngSanitize module defines ng-bind-html directive.
  var module = angular.module('GameDirectives', ['ngSanitize']);

  /**
   * Randomize array element order in-place
   * using Fisher-Yates shuffle algorithm.
   */
  function shuffleArray(array) {
    for (var i = array.length - 1; i > 0; i--) {
      var j = Math.floor(Math.random() * (i + 1));
      var temp = array[i];
      array[i] = array[j];
      array[j] = temp;
    }
    return array;
  }

  var deck = [];
  var ranks = [2, 3, 4, 5, 6, 7, 8, 9, 10, 'J', 'Q', 'K', 'A'];
  var suits = ['spades', 'hearts', 'diamonds', 'clubs'];
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Playing Cards ...

playing-card.js

```
module.factory('playingCardSvc', function () {
  var svc = {};

  svc.dealCard = function () {
    return deck.pop();
  };

  svc.dealHand = function (count) {
    var hand = [];
    for (var i = 0; i < count; i++) {
      var card = svc.dealCard();
      if (card) hand.push(card);
    }
    return hand;
  };

  svc.newDeck = function () {
    deck = [];
    suits.forEach(function (suit) {
      ranks.forEach(function (rank) {
        deck.push({rank: rank, suit: suit});
      });
    });
    shuffleArray(deck);
  };

  svc.newDeck();

  return svc;
});
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

… Playing Cards

playing-card.js

```
module.directive('playingCard', function () {
  return {
    // ng-bind-html avoids escaping content
    // so character entities can be displayed.
    // Those are used for suit characters.
    template: '<div class="playing-card" style="color:{{color}}"' +
      ' ng-bind-html="content"></div>',
    replace: true,
    scope: {
      playingCard: '='
    },
    link: function (scope) {
      var suit = scope.playingCard.suit;
      scope.color = suit === 'hearts' || suit === 'diamonds' ?
        'red' : 'black';
      if (suit === 'diamonds') suit = 'diams'; // unicode name

      scope.content = scope.playingCard.rank +
        '<br>&' + suit + ';';
    }
  };
});
})();
```

`playingCard` is an object with `rank` and `suit` properties

see http://en.wikipedia.org/wiki/Playing_cards_in_Unicode

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Event Handling Directives

- Examples
  - restrict keys that can be pressed when focus is in an `input` element
  - change styling or content of an element when mouse hovers over it

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Event Handling ...

Age 52
Age is 52.
Hover over me!

index.html

```
<!DOCTYPE html>
<html ng-app="Demo">
  <head>
    <link rel="stylesheet" href="demo.css">
    <script src=".../angular.min.js"></script>
    <script src="event-directives.js"></script>
    <script src="demo.js"></script>
  </head>
  <body ng-controller="DemoCtrl">
    <label>Age</label>
    <input type="text" digits-only ng-model="age" autofocus>
    <!-- Could also use type="number". -->
    <div>Age is {{age}}.</div>
    <div class="box" hover-color="pink">Hover over me!</div>
  </body>
</html>
```

demo.css

```
body {
  font-family:
    sans-serif;
}

.box {
  background-color: yellow;
  border: solid black 1px;
  border-radius: 5px;
  margin: 5px;
  padding: 5px;
  width: 120px;
}
```

demo.js

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

app.controller('DemoCtrl', function ($scope) {
  // Just provides a scope for holding age.
});
```

otherwise **age** will be on root scope which would be fine for this small demo

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Event Handling ...

event-directives.js

```
(function () {
  'use strict';
  var module = angular.module('EventDirectives', []);

  // Checks for delete, tab, and arrow keys.
  function isNavigation(event) {
    var code = event.keyCode;
    return !event.shiftKey &&
      (code === 8 || code === 9 ||
      (code >= 37 && code <= 40));
  }

  // Checks for 0 to 9 keys.
  function isDigit(event) {
    var code = event.keyCode;
    return !event.shiftKey &&
      ((code >= 48 && code <= 57) ||
       (code >= 96 && code <= 105)); // keypad
  }
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Event Handling ...

```
/**
 * Restricts keys that can be pressed when an input has focus.
 * Example usage: <input type="number" digits-only>
 */
module.directive('digitsOnly', function () {
  return {
    link: function (scope, element, attrs) {
      element.on('keydown', function (event) {
        var valid = isDigit(event) || isNavigation(event);
        if (!valid && event.preventDefault) event.preventDefault();
        return valid; // for IE8 and earlier
      });
    }
  };
});
```

in old versions of IE, `Event` objects do not have the `preventDefault` method

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Event Handling

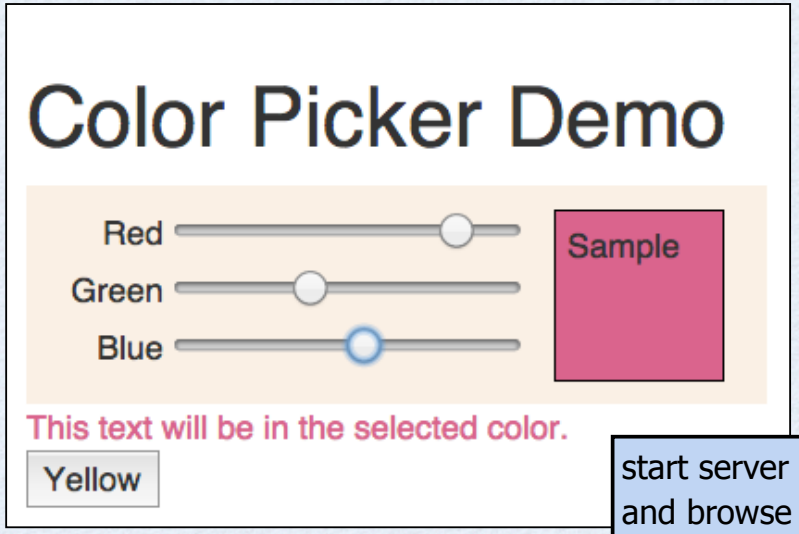
event-directives.js

```
  /**
   * Changes background color of an element
   * when mouse hovers over it.
   * Example usage: <div hover-color="pink">...</div>
   */
  module.directive('hoverColor', function () {
    return {
      scope: {
        hoverColor: '@'
      },
      link: function (scope, element, attrs) {
        var prevColor;
        element.on('mouseover', function (event) {
          prevColor = element.css('background-color');
          element.css('background-color', scope.hoverColor);
        });
        element.on('mouseout', function (event) {
          element.css('background-color', prevColor);
        });
      }
    };
  });
})();
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Using `ng-model`

- The provided directives **`input`**, **`select`** and **`textarea`** use the **`ng-model`** attribute to provide two-way data binding
- Custom widget directives can do this also



start server with "**`node server.js`**" and browse localhost:3000; necessary in Chrome to get around "cross-origin request" issue with directives loading template files from local file system

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Color Picker ...

color-picker.html

```
<div class="color-picker">
  <div class="color-picker-sliders">
    <div>
      <label>Red</label>
      <!-- range inputs are not supported yet
           in IE -->
      <input type="range" min="0" max="255"
        ng-model="color.red">
    </div>
    <div>
      <label>Green</label>
      <input type="range" min="0" max="255"
        ng-model="color.green">
    </div>
    <div>
      <label>Blue</label>
      <input type="range" min="0" max="255"
        ng-model="color.blue">
    </div>
  </div>
  <div class="color-picker-swatch"
    ng-style="swatchStyle">
    Sample
  </div>
</div>
```

color-picker.css

```
input {
  width: 150px;
}

label {
  display: inline-block;
  font-weight: normal;
  text-align: right;
  width: 50px;
}

.color-picker {
  width: 320px;
}

.color-picker-sliders {
  display: inline-block;
  margin: 10px;
}

.color-picker-swatch {
  border: solid black 1px;
  display: inline-block;
  height: 74px;
  padding: 5px;
  vertical-align: top;
  width: 74px;
}
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Color Picker ...

color-picker.js

```
(function () {
  'use strict';

  var module = angular.module('MyDirectives', []);

  module.factory('colorToRgbString', function () {
    return function (color) {
      return 'rgb(' + color.red + ',' +
        color.green + ',' + color.blue + ')';
    };
  });
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Color Picker ...

color-picker.js

```
  /**
   * Example usage:
   * <rmv-color-picker ng-model="myColor"></rmv-color-picker>
   */
  module.directive('rmvColorPicker', function (colorToRgbString) {
    function updateSwatch(scope) {
      scope.swatchStyle.backgroundColor = colorToRgbString(scope.color);
    }

    return {
      restrict: 'AE',
      templateUrl: 'color-picker.html',
      replace: true,
      scope: {
        color: '=ngModel'
      },
      link: function (scope, element) {
        scope.swatchStyle = {};
        var fn = updateSwatch.bind(null, scope);
        scope.$watch('color.red', fn);
        scope.$watch('color.green', fn);
        scope.$watch('color.blue', fn);
      }
    };
  });
})();
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

… Color Picker …

index.html

```
<!DOCTYPE html>
<html ng-app="Demo">
  <head>
    <title>AngularJS color-picker directive</title>
    <link rel="stylesheet" href="color-picker.css">
    <link rel="stylesheet" href="demo.css">
    <script src=".../angular.min.js"></script>
    <script src="color-picker.js"></script>
    <script src="demo.js"></script>
  </head>
  <body ng-controller="DemoCtrl">
    <h1>Color Picker Demo</h1>
    <rmv-color-picker ng-model="myColor">
    </rmv-color-picker>
    <div ng-style="textStyle">
      This text will be in the selected color.
    </div>
    <button ng-click="goToYellow()">Yellow</button>
  </body>
</html>
```

demo.css

```
body {
  font-family: sans-serif;
  padding: 10px;
}
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

… Color Picker

demo.js

```
(function () {
  'use strict';

  var app = angular.module('Demo', ['MyDirectives']);

  app.controller('DemoCtrl', function ($scope, colorToRgbString) {
    // Initial values.
    $scope.myColor = {red: 127, green: 127, blue: 127}; // gray
    $scope.textStyle = {};

    $scope.goToYellow = function () {
      $scope.myColor = {red: 255, green: 255, blue: 0};
    };

    $scope.$watchCollection('myColor', function (color) {
      $scope.textStyle.color = colorToRgbString(color);
    });
  });
})();
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Link Function Parameters

- **`scope`**
  - scope of this directive
  - can be shared, inherited or isolate
- **`element`**
  - jqLite or jQuery wrapped DOM element on which this directive appears
- **`attrs`**
  - map of attributes on the DOM element where directive appears
- **`requiredControllers`**
  - not commonly used
  - described on next slide

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Requiring Other Directives

not commonly used

- A directive can "require" the presence of other directives on the same element or ancestor elements
  - using directive definition object property **`require`**
    - set to a string directive name for one or an array of string names for multiple
  - prefix names with "^" if directive can appear on an ancestor element, otherwise must be on same element
  - controllers of required directives are passed to the **`link`** function in 4th parameter, typically named **`requiredControllers`**
    - value will be a single controller object if the **`require`** value was a string, or an array of controller objects if it was an array
    - can call methods on these controllers that are defined with **`this.name = fn;`**
- Provides ability to communicate between directives
  - can call methods defined on controller object with ***`ctrlObj.name`***`();` as opposed to calling methods defined on scope of controller with **`$scope.`***`name`***`()`

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

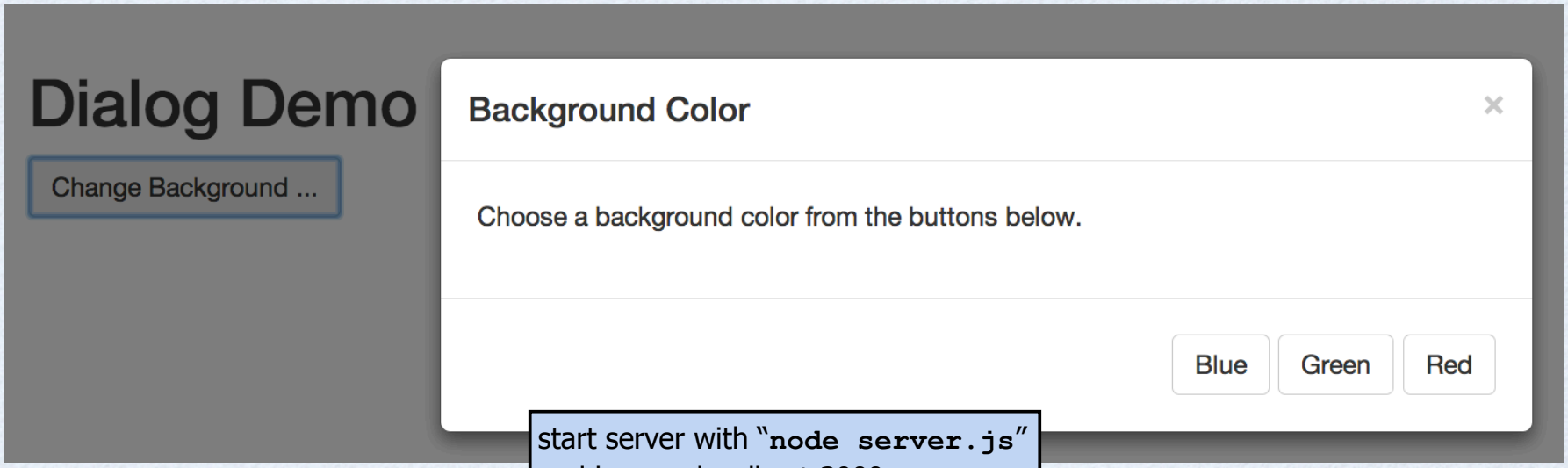
Transclusion

- Element on which directive appears
  can have content that is inserted into the directive template
  in a location that it chooses
  - content can use other directives

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Transclusion Example

- Directive that wraps the Twitter Bootstrap modal widget
  - http://getbootstrap.com/javascript/#modals
  - Twitter Bootstrap depends on jQuery
- Can specify header title, body content, and footer buttons



start server with "`node server.js`" and browse localhost:3000; necessary in Chrome to get around "cross-origin request" issue with directives loading template files from local file system

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Dialog ...

dialog.js

```
(function () {
  'use strict';

  var module = angular.module('MyDirectives', []);

  /**
   * Example usage:
   * <rmv-dialog title="Make a Move" btn-map="btnMap">
   *   ... content goes here ...
   * </rmv-dialog>
   * where btnMap is an object on the scope
   * whose keys are the text for buttons in the footer and
   * whose values are functions to invoke when the buttons are pressed.
   * Omit btn-map if no buttons are needed.
   * In that case there will be no footer area.
   */
  module.directive('rmvDialog', function () {
    return {
      restrict: 'AE',
      templateUrl: 'dialog.html',
      replace: true,
      transclude: true,
      scope: {
        btnMap: '=',
        title: '@'
      }
    };
  });
})();
```

enables usage of transclusion

if `'element'` is specified instead of `true`, the element on which the directive appears is part of the transcluded content, not just its content

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Dialog ...

dialog.html

```
<div class="modal fade">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button class="close" data-dismiss="modal">&times;</button>
        <h4 class="modal-title">{{title}}</h4>
      </div>
      <div class="modal-body" ng-transclude></div>
      <div class="modal-footer" ng-show="btnMap">
        <button type="button" class="btn btn-default" data-dismiss="modal"
          ng-repeat="(text, fn) in btnMap" ng-click="fn()">
          {{text}}
        </button>
      </div> <!-- modal-footer -->
    </div> <!-- modal-content -->
  </div> <!-- modal-dialog -->
</div> <!-- modal -->
```

marks where content will go

in addition to executing the associated function, each button also dismisses the dialog

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

… Dialog …

index.html

```
<html ng-app="Demo">
  <head>
    <title>AngularJS dialog directive</title>
    <link rel="stylesheet"
      href="http://netdna.bootstrapcdn.com/bootstrap/3.1.0/css/bootstrap.min.css">
    <link rel="stylesheet" href="demo.css">
    <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
    <script src="http://netdna.bootstrapcdn.com/bootstrap/3.1.0/js/bootstrap.min.js">

    </script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.21/angular.min.js">
    </script>
    <script src="demo.js"></script>
    <script src="dialog.js"></script>
  </head>
  <body ng-controller="DemoCtrl" ng-style="bodyStyle">
    <h1>Dialog Demo</h1>
    <button class="btn btn-default" data-toggle="modal" data-target="#myDialog">
      Change Background ...
    </button>
    <rmv-dialog id="myDialog" title="Background Color" btn-map="myBtnMap">
      Choose a background color from the buttons below.
    </rmv-dialog>
  </body>
</html>
```

demo.css

```
body {
  padding: 15px;
}
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

... Dialog

demo.js

```
(function () {
  'use strict';

  var app = angular.module('Demo', ['MyDirectives']);

  app.controller('DemoCtrl', function ($scope) {
    function setBgColor(color) {
      $scope.bodyStyle = {backgroundColor: color};
    }

    $scope.myBtnMap = {
      'Red': function () { setBgColor('red'); },
      'Green': function () { setBgColor('green'); },
      'Blue': setBgColor.bind(null, 'blue') // different way to write
    };
  });
})();
```

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

$destroy Event

- Some directives require cleanup to avoid leaking memory
- Examples include
  - unregistering listeners on scope and DOM events
  - stopping periodic execution of functions initiated with the **$interval** service
- A directive instance is "destroyed" when its element is removed from the DOM or when the page is closed in the browser
- Directive **link** functions can register listeners for this

```
scope.$on('$destroy', function () {
  // perform non-DOM cleanup here
});
```

```
element.on('$destroy', function () {
  // perform DOM cleanup here
});
```

element is a jqLite or jQuery wrapped element

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Recursive Directives

- Useful when the template needs to be generated at run-time based on data in the app
- Useful when the number of DOM elements needed in the template varies based on the amount of data to be rendered
  - ex. tree-like data

- Can be implement using **`$compile`** service
- **`$compile(`*`template`*`)`**
  - ***`template`*** can be an HTML string (most common) or a DOM element
  - returns a function that should be invoked with a scope object and another function
  - the other function is passed a DOM element that **`$compile`** created from the template
  - this DOM element can be appended to the element on which the directive was applied
  - got that? no? example on next two slides will help

In the case that the template was specified as a DOM element, this DOM element is a clone of it. That's why the name of that parameter in the documentation is **`clonedElement`**.

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Recursive Example ...

demo.js

```
<!DOCTYPE html>
<html ng-app="Demo">
  <head>
    <script src=".../angular.min.js"></script>
    <script src="demo.js"></script>
  </head>
  <body ng-controller="DemoCtrl">
    <h3>Object Inspector</h3>
    <inspect value="myObj"></inspect>
  </body>
</html>
```

demo.js

```
'use strict';
var app = angular.module('Demo', []);

app.controller('DemoCtrl', function ($scope) {
  $scope.myObj = {
    never: undefined,
    none: null,
    bool: true,
    number: 19,
    text: 'some text',
    arr: [1, 3, 7],
    level1: {
      level2: {
        level3: {
          level4: 'end'
        }
      }
    }
  };
});
```

Object Inspector

- arr
  - 0 = 1
  - 1 = 3
  - 2 = 7
- bool = true
- level1
  - level2
    - level3
      - level4 = "end"
- never = undefined
- none = null
- number = 19
- text = "some text"

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

… Recursive Example

demo.js

```
app.directive('inspect', function ($compile) {
  return {
    restrict: 'E',
    replace: true,
    scope: {
      value: '='
    },
    link: function (scope, element) {
      var quote, template, v = scope.value, type = typeof v;

      if (v && type === 'object') { // object or array, not null
        template = '<ul><li ng-repeat="(k, v) in value">{{k}}' +
          '<inspect value="v"></inspect></li></ul>';
      } else {
        quote = type === 'string' ? '"' : '';
        template = '<span> = ' + quote + v + quote + '</span>';
      }

      $compile(template)(scope, function (clonedElement) {
        element.append(clonedElement);
      });
    }
  };
});
```

recursion!

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved

Compile Function

advanced feature

- Used to manipulate the DOM AFTER the **`link`** function runs to add elements to and remove elements from the DOM
  - ex. to use a template more than once
  - ex. to repeat an element based on changes to a scope property
- Important so binding expressions get evaluated before operating on the view in the **`link`** function
- Most directives don't need this
- No scope is available here
- Provided directives that do this include
  - **`ng-if`**, **`ng-switch`** and **`ng-repeat`**

Copyright © 2013-2014 by Object Computing, Inc. (OCI)
All rights reserved