CUSTOM DIRECTIVES

http://oricalvo.com
ori.calvo@gmail.com

Objectives

- Understanding directive lifecycle
- Implement custom directive
- Review all directive's options
- Taking full control of directing compilation and linking phase

http://oricalvo.com
ori.calvo@gmail.com

Integrating jQuery UI datepicker

- Suppose we hold the following HTML

```
<div class="home-view" ng-controller="HomeCtrl">
     <h1>Home</h1>
     Date:
     <input type="text" class="datepicker"/>
</div>
```

- And would like to transform the input field into jQuery UI datepicker
- We need to invoke jQuery UI datepicker function on the input DOM element
- How can we get access to the DOM element being managed by the controller?

http://oricalvo.com
ori.calvo@gmail.com

Datepicker – Solution 1

- Use $element

```
angular.module("myApp").controller("HomeCtrl", function ($scope, $element) {
    $element.find(".datepicker").datepicker();
});
```

- But this means that the controller is coupled to the HTML
- When running under unit test we must provide a reference to the DOM element
- Breaks controller testability

Datepicker – Solution 2

- A better solution is to implement a **directive**
- We can define a new directive named datepicker
- Put inside it all the DOM manipulation logic
- Let Angular initialize it and attach it to the DOM
- HTML becomes cleaner ☺
- Controller is free from DOM housekeeping

```
<div class="home-view" ng-controller="HomeCtrl">
     <h1>Home</h1>
     Date:
     <datepicker />
</div>
```

Directive

- Directive can appear as
  - Element
  - Attribute
  - Comment
  - CSS Class

```
<body class="my-directive">
     <my-directive></my-directive>
     <div class="home-view" ng-controller="HomeCtrl" my-directive></div>
     <!-- directive: my-directive exp -->
     <div></div>
</body>
```

http://oricalvo.com
ori.calvo@gmail.com

Custom Directive

- A directive must be registered with a module
  - Name
  - Function
- The function must return a directive definition object

```
angular.module("myApp").directive("datepicker", function () {
    return {
        link: function (scope, element, attrs) {
        }
    };
});
```

In most cases implementing just a link function is good enough

restrict

- By default Angular matches a directive using HTML attribute or element

```
<div datepicker />
```

```
<datepicker />
```

- Use restrict property to change default

```
angular.module("myApp").directive("datepicker", function () {
    return {
        restrict: "E",
        link: function (scope, element, attrs) {
            element.datepicker();
        },
    };
});
```

- Supported values are: A, E, C, M

Directive Factory

- Angular might invoke the directive function even if no matching element/attribute/class/comment is found

Directive is instantiated before angular checks that restrict option

```
function addDirective(tDirectives, name, location, ...) {
     var match = null;
     if (hasDirectives.hasOwnProperty(name)) {
         for (var directive, directives = $injector.get(name + Suffix),
              i = 0, ii = directives.length; i < ii; i++) {
             try {
                 directive = directives[i];
                 if (directive.restrict.indexOf(location) != -1) {
                     tDirectives.push(directive);
                     match = directive;
                 }
             } catch (e) { $exceptionHandler(e); }
         }
     }
     return match;
}
```

http://oricalvo.com
ori.calvo@gmail.com

replace & template

- jQuery UI datepicker accepts a DOM element of type input
- However, we want to support the following HTML

```
<datepicker />
```

- Use replace + template

```
angular.module("myApp").directive("datepicker", function () {
     return {
         restrict: "E",
         link: function (scope, element, attrs) {
             element.datepicker();
         },
         replace: true,
         template: '<input type="text" />‘
     };
});
```

replace & template

```
if (directive.template) {
    directiveValue = (isFunction(directive.template))
         ? directive.template($compileNode, templateAttrs)
         : directive.template;

     directiveValue = denormalizeTemplate(directiveValue);

     if (directive.replace) {
         replaceDirective = directive;
         $template = removeComments(wrapTemplate(directive.templateNamespace, trim(directiveValue)));
         compileNode = $template[0];

         replaceWith(jqCollection, $compileNode, compileNode);

         var newTemplateAttrs = { $attr: {} };
         var templateDirectives = collectDirectives(compileNode, [], newTemplateAttrs);
         var unprocessedDirectives = directives.splice(i + 1, directives.length - (i + 1));
         directives = directives.concat(templateDirectives).concat(unprocessedDirectives);
         mergeTemplateAttributes(templateAttrs, newTemplateAttrs);
         ii = directives.length;
     } else {
         $compileNode.html(directiveValue);
     }
}
```

$template is a jQuery object which represents the tamplate content

Replace current element being compiled with new template node

Merge current element's directives with directives from the template

http://oricalvo.com
ori.calvo@gmail.com

replace is Deprecated

- According to Angular docs the replace option is deprecated
- However, according to many questions being asked, the option will not be removed easily
- More info can be found at https://github.com/angular/angular.js/commit/eec6394a342fb92fba5270eee11c83f1d895e9fb
- Angular 1.4 still supports it

templateUrl

- Directive's template may be complex
- You may extract the template into a separate HTML file and let Angular download it on demand

```
angular.module("myApp").directive("datepicker", function () {
     return {
         restrict: "E",
         link: function (scope, element, attrs) {
             element.datepicker();
         },
         replace: true,
         templateUrl: "/views/Main/DatepickerTemplate“
     };
});
```

```
<input type="text" class=".datepicker" />
```

http://oricalvo.com
ori.calvo@gmail.com

Delayed Compilation

- Browser needs to download the template from the server
- Compilation and linking is suspended until download completes
- Parent scope sees original HTML
- Compilation is delayed even if template is present using text/ng-template ☹

Directive Controller

- Directive with complex template can be managed by a dedicated controller

```
angular.module("myApp").directive("clock", function () {
     return {
         restrict: "E",
         templateUrl: "/views/Main/ClockTemplate",
         controller: ClockDirectiveController,
     };
});

function ClockDirectiveController($scope) {
      $scope.time = new Date();
}
```

- Angular calls it “Component”

http://oricalvo.com
ori.calvo@gmail.com

compile vs. link. vs. controller

- Directive's controller is instantiated before pre linking phase
- This implies that one directive's preLink may ask for parent directive's controller

```
<div ng-controller="HomeCtrl">
     <parent>
          <child></child>
     </parent>
</div>
```

http://oricalvo.com
ori.calvo@gmail.com

Accessing Child Controllers

- According to previous slide controller's constructor is invoked before child controllers are created
- Think about the ng-model directive
  - It binds itself to the parent scope (through the form controller)
  - You may want to interact with ngModelController instance inside the parent controller
  - However, no ngModelController instance is created yet ☹

http://oricalvo.com
ori.calvo@gmail.com

Directive Scope

- By default Angular does not create a new scope for the directive
- The directive might corrupt data used by the outer scope
  - Or use data that is not intended for it
- This is unacceptable for a directive that is used as a component
  - Should be isolated and encapsulated

Directive Scope

```
angular.module("myApp").controller("HomeCtrl", function ($scope) {
    $scope.title = "Home";
});
```

```
angular.module("myApp").directive("clock", function ($parse) {
     return {
         restrict: "E",
         replace: true,
         templateUrl: "/views/Main/Clock",
         controller: ClockDirectiveController,
    };
});
```

By default, a directive does not create a new scope

$scope is the scope of HomeCtrl not Clock directive

```
function ClockDirectiveController($scope) {
     this.time = new Date();
     $scope.title = "Clock";
}
```

Directive which Creates a Scope

- You can ask Angular to create a dedicated scope for your directive
- scope: true → Derived scope. New scope which inherits from the parent scope
- scope: {} → Isolated scope. Can still use $parent to access parent scope

```
angular.module("myApp").directive("clock", function ($parse) {
     return {
         scope: true,
         //scope: {},
         restrict: "E",
         replace: true,
         templateUrl: "/views/Main/Clock",
         controller: ClockDirectiveController,
         controllerAs: "ctrl",
     };
});
```

http://oricalvo.com
ori.calvo@gmail.com

Isolated Scope

- Being isolated means that even sibling directives do not get access to the isolated scope
  - This is not true for inherited scope
- This behavior might be confusing
  - A DIV with both ng-controller and ng-show → Both directives share the scope
  - A DIV with my-isolated-directive and ng-show → ng-show is linked to the parent scope

```
<div my-dir ng-show="active">
     If you see this content then ng-show is linked to my-dir scope
</div>
```

You cannot determine from the HTML to which scope ng-show is linked

http://oricalvo.com
ori.calvo@gmail.com

Directive Controller as API

- The directive controller can be used as the public API of the directive
- The trick is to store the directive's controller inside the outer scope

```
<div class="home-view" ng-controller="HomeCtrl">
     <clock name="clock"></clock>
     <button ng-click="clock.move()">Inc</button>
</div>
```

```
function ClockDirectiveController($scope, $attrs, $parse) {
     this.$scope = $scope;
     if ($attrs.name) {
         var getter = $parse($attrs.name);
         var setter = getter.assign;
         if (!setter) {
             throw new Error(attrs.Name + " cannot be assigned");
         }
         setter($scope.$parent, this);
     }

     $scope.time = new Date();
}

ClockDirectiveController.prototype.move = function () {
     this.$scope.time.setHours(this.$scope.time.getHours() + 1);
}
```

Register a reference to the directive inside the parent scope

Directive's Events

- Continuing our clock directive
- It would be nice if the directive exposes an "onTick" event which the outer scope can register to

```
<div ng-controller="HomeCtrl">
     <clock ontick="tick(time)"></clock>
</div>
```

```
function ClockDirectiveController($scope, $attrs) {
     if ($attrs.ontick) {
         setInterval(function () {
             $scope.$apply(function () {
                 $scope.$parent.$eval($attrs.ontick, { time: new Date() });
             });
         }, 1000);
     }
}
```

Can send parameters to the event handler

```
angular.module("MyApp").directive("clock", function () {
     return {
         controller: ClockDirectiveController,
         templateUrl: "/Home/Clock",
         scope: {},
     };
});
```

& Symbol

- Indicates a method invocation expression
- The expression is parsed by angular and is made available on the directive's scope

```
function ClockDirectiveController($scope) {
    setInterval(function () {
        $scope.$apply(function () {
            $scope.tick({ time: new Date() });
        });
    }, 1000);
}

angular.module("MyApp").directive("clock", function () {
    return {
        controller: ClockDirectiveController,
        templateUrl: "/Home/Clock",
        scope: {
            tick: "&ontick",
        },
    };
});
```

```
<div ng-controller="HomeCtrl">
    <clock ontick="tick(time)"></clock>
</div>
```

tick is private while ontick is public

Can use tick: "&" instead

Directive's Properties

- You may want to bind directive property to outer scope data
- The directive may watch the data and update itself automatically

```
<div ng-controller="HomeCtrl">
     <clock time="time"></clock>
</div>
```

```
function HomeCtrl($scope, $interval) {
     $scope.time = new Date();

     $interval(function () {
         $scope.time = new Date();
     }, 1000);
}
```

```
function ClockDirectiveController($scope, $parse, $attrs) {
     if ($attrs.time) {
         $scope.$parent.$watch($attrs.time, function (newValue) {
             $scope.time = newValue;
         });
     }
}
```

Assuming isolated scope

= Symbol

- Indicates an expression which references a field
- Angular keeps the expression and the scope in sync

Are synched

```
function ClockDirectiveController($scope, $parse, $attrs) {
}

angular.module("MyApp").directive("clock", function () {
     return {
          controller: ClockDirectiveController,
          templateUrl: "/Home/Clock",
          scope: {
               time: "=",
          },
     };
});
```

```
function HomeCtrl($scope, $interval) {
     $scope.time = new Date();

     $interval(function () {
          $scope.time = new Date();
     }, 1000);
}
```

```
<div ng-controller="HomeCtrl">
     <clock time="time"></clock>
</div>
```

Interpolated Property

- In some cases the outer scope may want to inject a string into directive's property

```
<div ng-controller="HomeCtrl">
     <clock title="Hello, {{name}}"></clock>
</div>
```

```
function HomeCtrl($scope, $interval) {
     $scope.name = "MyClock";
     var counter = 0;
     $interval(function () {
         $scope.name = ("MyClock" + ++counter);
     }, 1000);
}
```

```
function ClockDirectiveController($scope, $attrs, $interpolate) {
     if ($attrs.title) {
         var strFn = $interpolate($attrs.title);
         $scope.$parent.$watch(
             function () {
                 return strFn($scope.$parent);
             }, function (newValue) {
                 $scope.title = newValue;
             });
     }
}
```

@ Symbol

- Indicates an expression with curly braces {{}}
- Angular updates directive's scope with "toString" of the expression

```
function ClockDirectiveController($scope, $attrs, $interpolate) {
}

angular.module("MyApp").directive("clock", function () {
     return {
          controller: ClockDirectiveController,
          templateUrl: "/Home/Clock",
          scope: {
               title: "@",
          },
     };
});
```

```
<div ng-controller="HomeCtrl">
     <clock title="Hello, {{name}}"></clock>
</div>
```

```
<div>
     <h1>{{title}}</h1>
     Time is: <span>{{time | date : 'mediumTime'}}</span>
</div>
```

http://oricalvo.com
ori.calvo@gmail.com

bindToController

- When implementing Component you may find it useful that scope bindings are applied to the controller rather to the scope

```
angular.module("MyApp").directive("clock", function () {
    return {
        controller: ClockDirectiveController,
        templateUrl: "/Home/Clock",
        bindToController: true,
        controllerAs: "ctrl",
        scope: {
            time: "=",
        },
    };
});
```

```
<div ng-controller="HomeCtrl">
    <clock time="time"></clock>
</div>
```

```
function HomeCtrl($scope, $interval) {
    $interval(function () {
        $scope.time = new Date();
    }, 1000);
}
```

```
<div>
    <h1>Clock</h1>
    Time is: <span>{{ctrl.time}}</span>
</div>
```

Outer controller's time is bound to Directive's controller's time

Require Other Directive

- A directive may specify a list of other directive's controllers dependencies
- For example, ngModel has optional dependency on form directive

```
var ngModelDirective = ['$rootScope', function ($rootScope) {
     return {
         restrict: 'A',
         require: ['ngModel', '^?form', '^?ngModelOptions'],
         controller: NgModelController,
         priority: 1,
         compile: function ngModelCompile(element) {
             return {
                 pre: function ngModelPreLink(scope, element, attr, ctrls) {
                     var modelCtrl = ctrls[0],
                         formCtrl = ctrls[1] || nullFormCtrl;
                 },
                 post: function ngModelPostLink(scope, element, attr, ctrls) {
                 }
             };
         }
     };
 }];
```

http://oricalvo.com
ori.calvo@gmail.com

Require

- Can be a single string or an array of strings
- Each dependency can be attributed with
  - No prefix – Locate the required controller on the current element. Throws an error if controller was not found
  - ? – Optional dependency
  - ^ - Search current element and parents
  - ^^ - Search only parents
  - ?^, ?^^ - Optional dependency

Require

```
angular.module("MyApp").directive("page", function () {
    return {
        controller: PageDirectiveController,
        require: ["page", "^^tab"],
        scope: {},
        link: {
            pre: function (scope, element, attrs, ctrls) {
                var pageCtrl = ctrls[0];
                var tabCtrl = ctrls[1];

                ...

                tabCtrl.addPage(pageCtrl);
            },
            post: function () {
            }
        }
    };
});
```

```
angular.module("MyApp").directive("tab", function () {
    return {
        controller: TabDirectiveController,
        link: function (scope, element, attrs, ctrl) {
            ...
        }
    };
});
```

```
function TabDirectiveController($scope) {
    this.pages = {};
}

TabDirectiveController.prototype.addPage = function (page) {
    this.pages[page.name] = page;
}
```

```
<tab>
    <page name="page1" title="Page 1">
        ...
    </page>
    <page name="page2" title="Page 2">
        ...
    </page>
</tab>
```

Directive with Arbitrary Content

- A tab contains a header with all pages
- The header can be injected during link phase using plain DOM manipulation

```
TabDirective.prototype.buildHeader = function () {
    var me = this;
    var header = angular.element("<div class='header'/>");
    this.element.prepend(header);
    for (var pageName in me.ctrl.pages) {
        var page = me.ctrl.pages[pageName];
        this.buildTitle(header, page);
        me.currentPage = me.currentPage || page;
    }

    if (me.currentPage) {
        me.currentPage.element.removeClass("ng-hide");
    }
}
```

- A better approach is to use a template with the predefined HTML

http://oricalvo.com
ori.calvo@gmail.com

transclude

- When using the template option all directive's content is lost
- transclude option allows us for specifying a template while original content is injected into the template
- The injected content is linked to directive's parent scope

http://oricalvo.com
ori.calvo@gmail.com

transclude – Dialog Use Case

- Suppose we want to implement a dialog directive

```
<dialog title="Hello">
     <p>
         User defined content which can access scope {{name}}
     </p>
</dialog>
```

- The dialog element should be replaced by a div
- However, we need to keep the content

```
<div title="Hello">
     <p>
         User defined content which can access scope {{name}}
     </p>
</div>
```

transclude Option

- When set to true, the content of the directive will be injected into the directive's template under an element with an ng-transclude attribute

```
angular.module("myApp").directive("dialog", function () {
    return {
        restrict: "E",
        transclude: true,
        link: function (scope, element, attrs) {
            element.dialog();
        },
        template: "<div ng-transclude />",
        replace: true,
    };
});
```

```
<dialog title="Hello">
    <p>
        User defined content
    </p>
</dialog>
```

- The transcluded content is not bound to the directive's scope but rather to its outer scope

transcludeFn

- You might find ng-transclude too limited
  - For example, when the transcluded content need to be duplicated multiple times
  - You can take full control of the transclusion logic by using the 5th parameter of the link method

```
function postLink(scope, element, attrs, ctrl, transcludeFn) {
     element.find("span.footer").each(function () {
         var footer = $(this);
         transcludeFn(function (clone) {
             footer.append(clone);
         })
     });
}
```

transcludeFn's Scope

- By default transcludeFn is linked to a new scope which inherits from the directive's parent scope
- When duplicating transcluded content you probably want to link each copy to a different scope

```
link: function (scope, element, attrs, ctrl, transcludeFn) {
     for (var i = 0; i < 5; i++) {
         var scope = scope.$parent.$new();

         scope.num = i;

         transcludeFn(scope, function (clone) {
             element.append(clone);
         });
     }
}
```

http://oricalvo.com
ori.calvo@gmail.com

transclude: element

- Transcludes the whole directive's element
  - template option is ignored
- Original element is removed from the DOM
  - So what is sent to the link function ?

```
angular.module("MyApp").directive("panel", function () {
    return {
        transclude: "element",
        link: function (scope, element, attrs, ctrl, transcludeFn) {
            transcludeFn(function (clone, scope) {
                element.after(clone);
            });
        }
    };
});
```

Why use after and not append ?

```
<div ng-controller="HomeCtrl">
    <panel>
        <div>This content is transcluded {{num}}</div>
    </panel>
</div>
```

This is the actual DOM at runtime

```
<div ng-controller="HomeCtrl" class="ng-scope">
    <!-- panel: undefined -->
    <panel class="ng-scope">
        <div class="ng-binding">This content is transcluded 123</div>
    </panel>
</div>
```

http://oricalvo.com
ori.calvo@gmail.com

transclude: element

- Assuming multiple directives on the same element
- The highest priority directive is configured as transclude: element
- What about lower priority directives ? Do they have a chance to compile and link ?
  - Compilation is enforced immediately !!!
  - Linkage is postponed until transclusion is executed

http://oricalvo.com
ori.calvo@gmail.com

Compilation Conflicts

- Specifying two directives with transclusion on the same DOM node generates an error
- This is strange since both ngRepeat and ngIf use transclusion
  - No problem defining them both on the same element
  - BTW, who wins ?
- The answer: use directive.$$tlb option to signal Angular that multiple directives with transclusion is allowed

terminal – Taking full Control

- Building complex directive may require full control over compilation and linking phases
- Settings terminal to true means that angular should not recursively compile directive's child elements
- The directive instead uses Angular built-in services like $compile, $interpolate to do the work

```
angular.module("myApp").directive("grid", function ($interpolate, $compile) {
     return {
         restrict: "E",
         compile: function (element, attrs) {...},
         terminal: true,
    };
});
```

http://oricalvo.com
ori.calvo@gmail.com

Complex Directive - Grid

- We would like to support the following HTML

```
<grid grid-data="contacts" grid-data-item="contact">
     <column title="{{titles.ID}}">
         <span>{{contact.ID}}</span>
     </column>
     <column title="Name">
         <span>{{contact.Name}}</span>
     </column>
</grid>
```

```
angular.module("myApp").controller("HomeCtrl", function ($scope) {
     $scope.contacts = [
         { ID: 1, Name: "Ori" },
         { ID: 2, Name: "Roni" },
     ];

     $scope.titles = {
         ID: "ID",
    };
});
```

Grid - Definition

```
angular.module("myApp").directive("grid", function ($interpolate, $compile) {
    return {
        compile: function (element, attrs) {
            var metadata = GridDirective.compile($interpolate, $compile, element, attrs);

            return function postLink(scope, element, attrs) {
                var grid = new GridDirective(metadata);
                grid.link(scope, element, attrs);
            };
        },
        terminal: true,
        scope: {
            gridModel: "=",
        }
    };
});
```

http://oricalvo.com
ori.calvo@gmail.com

Grid - Compiling

```
GridDirective.compile = function ($interpolate, $compile, element, attrs) {
     var columns = [];

     element.find("column").each(function () {
         var column = $(this);

         var title = column.attr("title");
         if (title) {
             title = $interpolate(title);
         }

         var content = column.html();
         if (content) {
             content = $compile(content);
         }

         columns.push({
             title: title,
             content: content,
         });
     });

     element.replaceWith("<table><thead/><tbody/></table>");

     return {
         columns: columns,
     };
}
```

Grid - Linking

```
GridDirective.prototype.link = function (scope, element, attrs) {
     var me = this;

     me.scope = scope;
     me.element = element;
     me.attrs = attrs;

     me.buildHead();
     me.buildBody();
}
```

```
GridDirective.prototype.buildHead = function () {
     var me = this;

     var thead = me.table.find("thead");
     thead.empty();

     var tr = $("<tr/>").appendTo(thead);
     $.each(me.columns, function (index, column) {
         var th = $("<th/>").appendTo(tr);
         if (column.title) {
             var title = column.title(me.scope.$parent);
             th.text(title);
         }
     });
}
```

Grid – Linking (2)

```
GridDirective.prototype.buildBody = function () {
     var me = this;

     var tbody = me.table.find("tbody");
     tbody.empty();

     var data = me.scope.gridData;
     var dataItemName = me.attrs.gridDataItem;

     $.each(data, function (index, dataItem) {
         var childScope = me.scope.$parent.$new();
         childScope[dataItemName] = dataItem;

         var tr = $("<tr/>").appendTo(tbody);

         $.each(me.columns, function (index, column) {
             var td = $("<td/>").appendTo(tr);
             column.content(childScope, function (clone) {
                 td.append(clone);
             });
         });
     });
}
```

http://oricalvo.com
ori.calvo@gmail.com

Summary

- Complex SPA requires custom directive
- A good directive is simple to use
- And makes your JS/HTML cleaner
- Many directives creates a Domain Specific Language (DSL)
- Integrating Angular with 3rd party widget library requires implementing many directives
  - Can be tedious