:

Directive controller function

So far we've seen basics of directives in Angular. We saw how they can have a custom **template**, **isolated scope** and **link function**.

We saw that all the DOM manipulation the directive does plus probably scope access and manipulation is done within **link function** of the directive. In addition to this link function, directives can have **controller** functions. Controller function allows directives to communicate with each other and augment each other's behavior. This communication is actually done by means of another directive configuration property: **require**.

When your directive requires another directive it will get a hold of the required directive's controller as an extra argument to its **link** function. A directive can require other directive(s) whether on the same element, its immediate parent or one of its ancestors. See angular docs for more details. Here is an example of a directive **someDirective** which requires **someOtherDirective:**

Directives communication toy example

```
angular.module("demo", [])
 .directive("someDirective", someDirective)
 .directive("someOtherDirective", someOtherDirective);

function someDirective(){
 return{
  require: "someOtherDirective",
  link: function(scope, elem, attrs, someOtherDirectiveCtrl){
   someOtherDirectiveCtrl.exposedMethodForOthers();
  }
 }
}
function someOtherDirective(){
 return{
  controller: function(){
   this.exposedMethodForOthers = function(){
    console.log("I'm a method of someOtherDirective's controller");
   }
  }
 }
}
```

and in the HTML you may have something like this:

```
<div someDirective someOtherDirective>
</div>
```

To get a sense of why directives may need to communicate with each other, see tabs example from angular directive guide.

ngModel

ngModel is a directive with a useful controller which provides the essentials of **two way data bindings** to other directives. It also provide useful api for defining **validator**s on input components which use ngModel.

From angular docs on ngModel:

> *The ngModel directive binds an input,select, textarea (or custom form control) to a property on the scope using*

> *NgModelController, which is created and exposed by this directive.*

And from angular docs on ngModelController:

> *`NgModelController` provides API for the `ngModel` directive. The controller contains services for data-binding, validation, CSS updates, and value formatting and parsing. It purposefully does not contain any logic which deals with DOM rendering or listening to DOM events. Such DOM related logic should be provided by other directives which make use of `NgModelController` for data-binding to control elements. Angular provides this DOM logic for most `input` elements*

validators can be added to ngModelController using it's ***$validator*** property. **required** directive from angular.js is an example of using ngModelController for adding a validator (click **expand source** to see the code):

Angular required directive

```
var requiredDirective = function() {
  return {
    restrict: 'A',
    require: '?ngModel',
    link: function(scope, elm, attr, ctrl) {
      if (!ctrl) return;
      attr.required = true; // force truthy in case we are on non input
element

      ctrl.$validators.required = function(modelValue, viewValue) {
        return !attr.required || !ctrl.$isEmpty(viewValue);
      };

      attr.$observe('required', function() {
        ctrl.$validate();
      });
    }
  };
};
```

Exercises

Now we want to implement a couple of directives in order to see how communication of directives can be used in action.

Match validator

First thing we want to implement is a directive that communicates with ngModel and adds a validator which checks **equality to some other value.**

read more ...

Twitter bootstrap polyfills

A lot of Javascript part of the twitter bootstrap are somehow similar to how directives work in Angular. But it's a good practice not to use *bootstrap.js* directly when using Angular. There is a cool project named *ui.bootstrap* that contains a bunch of directives and services for bootstrap components written purely with AngularJs. It's highly recommended to use this module if you want to leverage the power of bootstrap in your application.

Even using *ui.bootstrap* you still might be tempted to add js part of Bootstrap to achieve some features which however can be implemented with minimum effort in pure angular.

In this section we want to implement a module containing some extremely simple (yet practically useful) directives which polyfills what you might be missing when using angular with css part of Bootstrap.

read more ...