Using Controllers Inside Directives

Step 1: Declare Controller in Directive

```
function MyDirective() {
  var ddo = {
    scope: {
      prop: '=',
    },
    controller: ControllerFunction,
    bindToController: true,
    controllerAs: 'myCtrl',
    templateUrl: 'template.html'
  };
  return ddo;
}
```

Attach declared scope properties to controller instance instead of directly to $scope

Use ‘myCtrl’ in directive’s template to refer to controller instance

Step 2: Define Controller

```
ControllerFunction.$inject = ['Service'];
function ControllerFunction(Service) {
  var myCtrl= this;

  myCtrl.method = function () {
    var name = "Hello " + myCtrl.prop;
    …
  };
}
```

Attach other properties to ‘this’ as usual

Use (& manipulate) props in isolate scope

Step 3: Use In Directive's Template

```
<div ng-if="myCtrl.method()">
 {{myCtrl.prop}}
</div>
```

Bi-Directional vs. One-way Binding

```
function MyDirective() {
  var ddo = {
    scope: {
      prop: '=',
    },
  ...
  };
  return ddo;
}
```

Bi-Directional vs. One-way Binding

```
function MyDirective() {
  var ddo = {
    scope: {
      prop: '<',
    },
  ...
  };
  return ddo;
}
```

One-way binding:
Watches only the identity of the parent property, not the property inside directive

Summary

- To add functionality to the directive, one choice is to use a controller that's declared directly on the DDO
- Use controller property to declare controller in DDO
- Use bindToController and controllerAs props to bind declared properties in isolate scope directly to controller instance
- Define controller function as usual
- Whenever possible, use '<' for one-way binding to save resources instead of bidirectional binding with '='