

@joedotjs

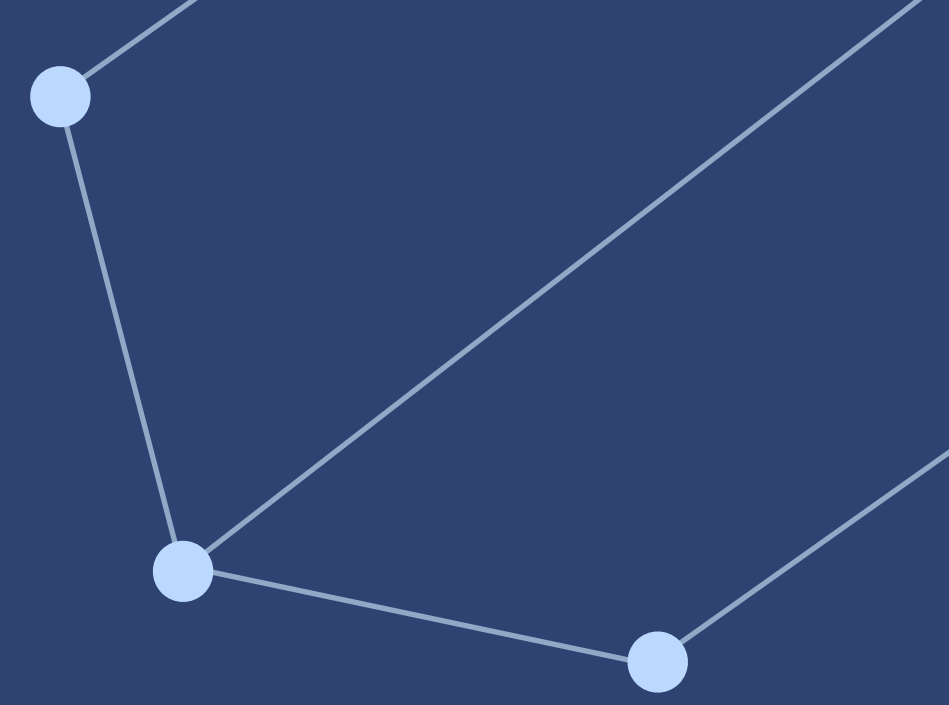




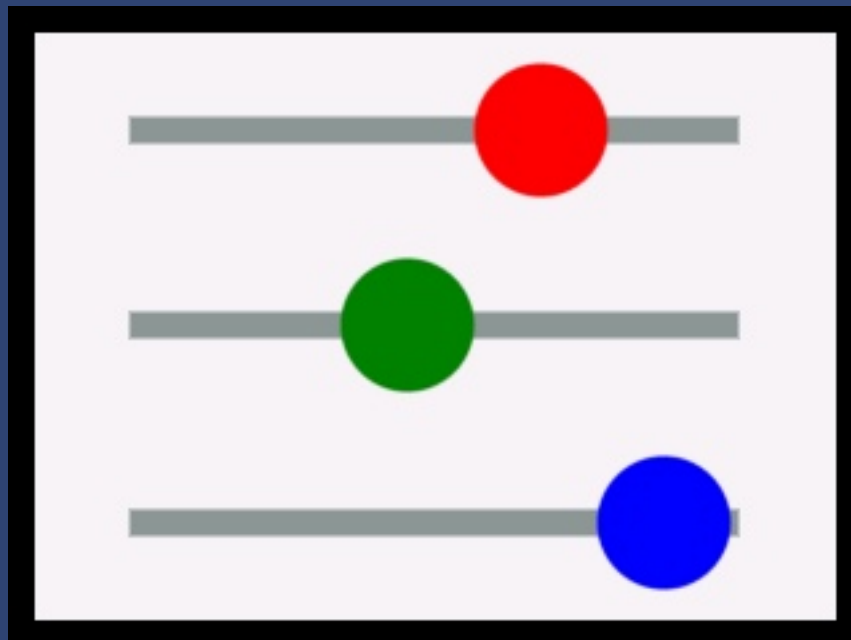
FULLSTACK
ACADEMY *of* CODE

Building Custom Inputs with **ngModelController**

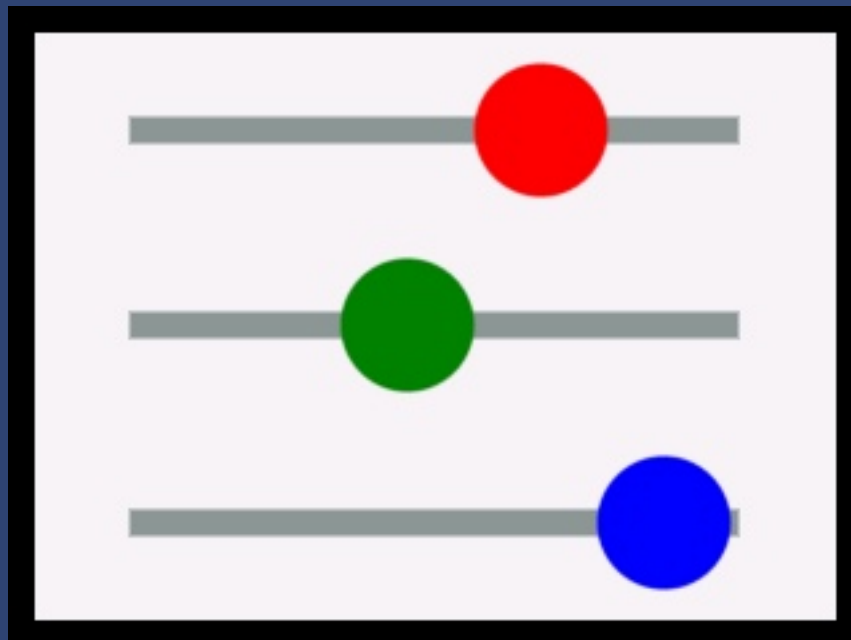
Directive-to-Directive Communication



<rgb-slider>



<rgb-slider>

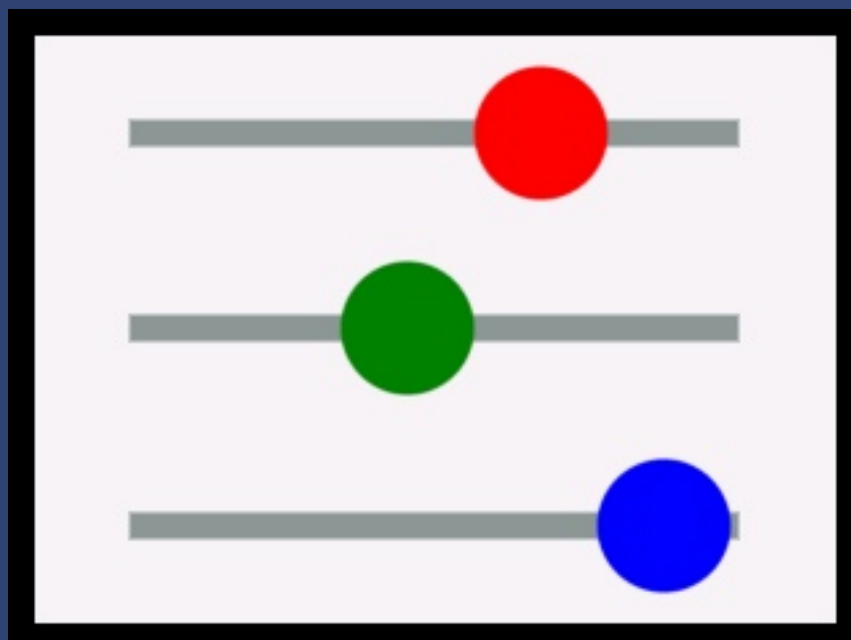


<color-palette>



\$scope

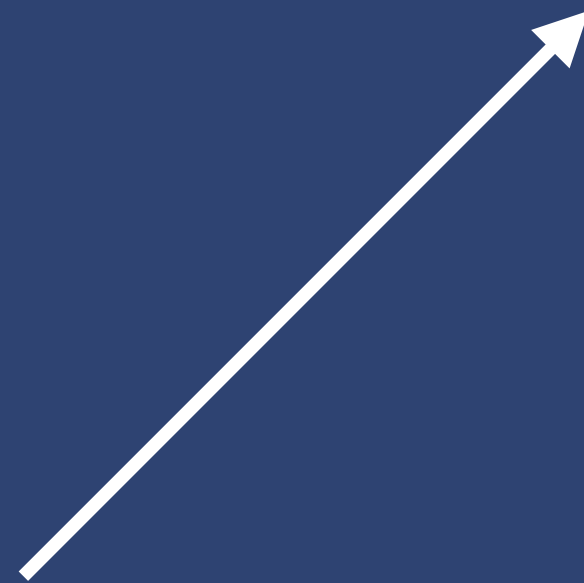
<rgb-slider>



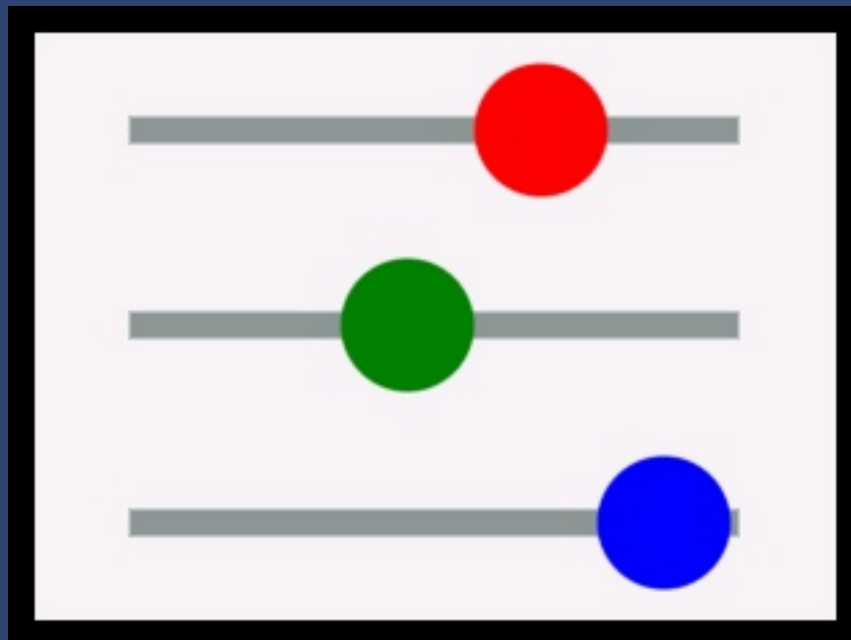
<color-palette>



\$scope



<rgb-slider>

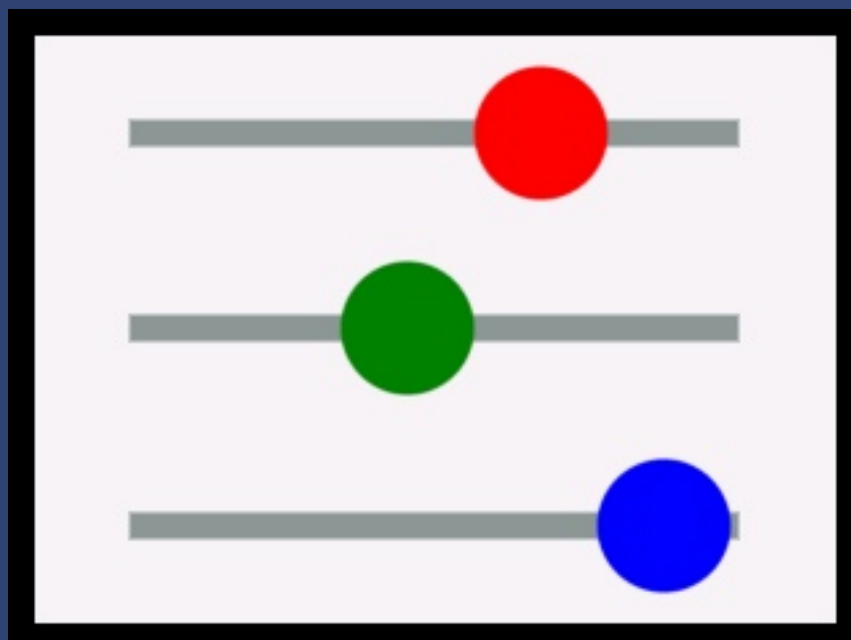


<color-palette>



\$scope

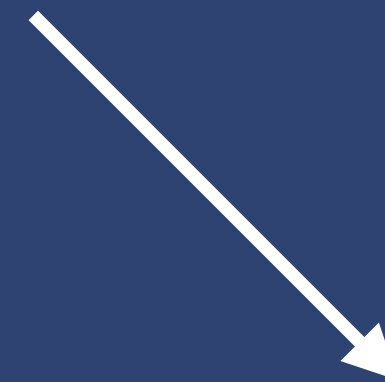
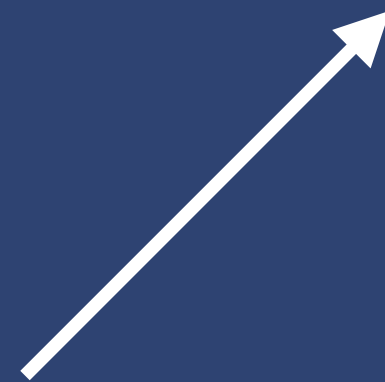
<rgb-slider>



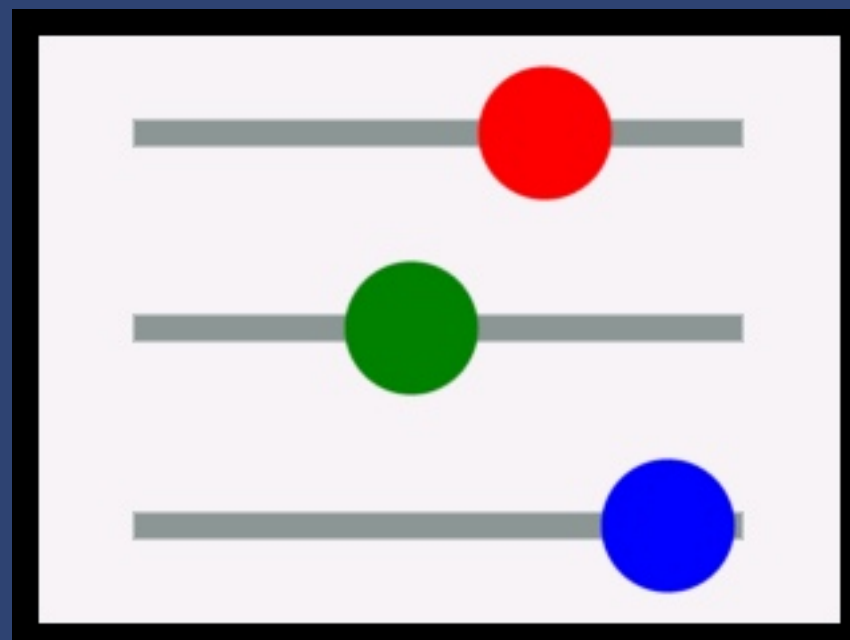
<color-palette>



\$scope



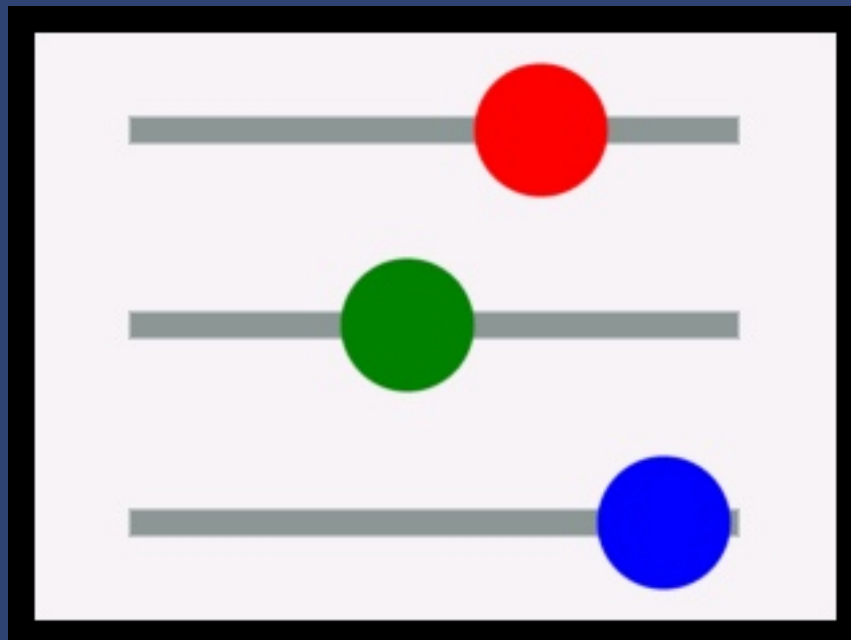
<rgb-slider>



<color-palette>



<rgb-slider>

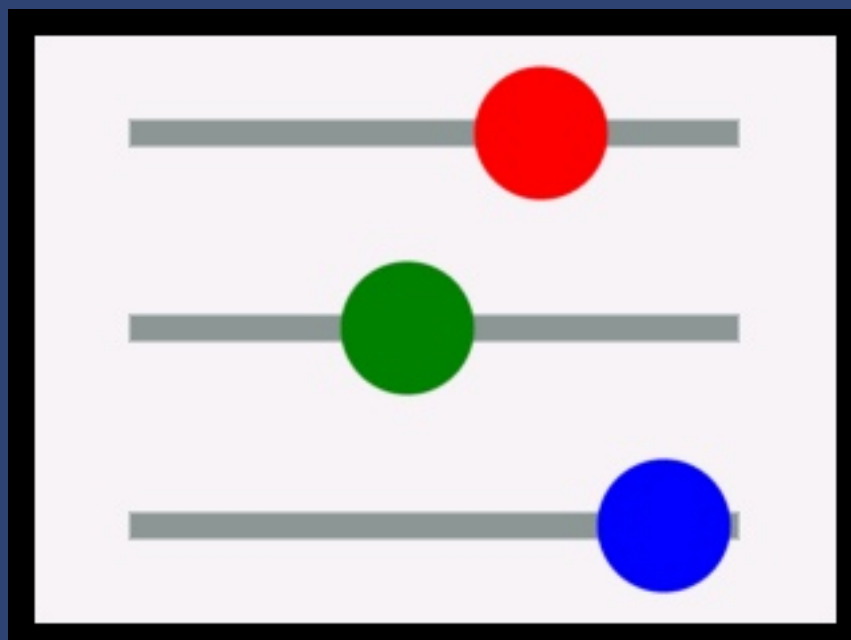


<color-palette>



ColorFactory

<rgb-slider>

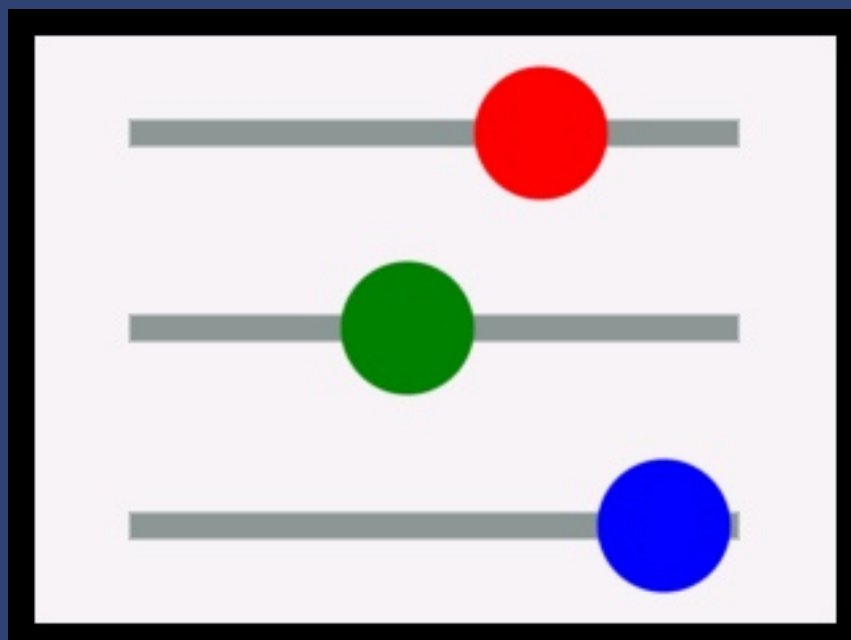


<color-palette>



ColorFactory

<rgb-slider>

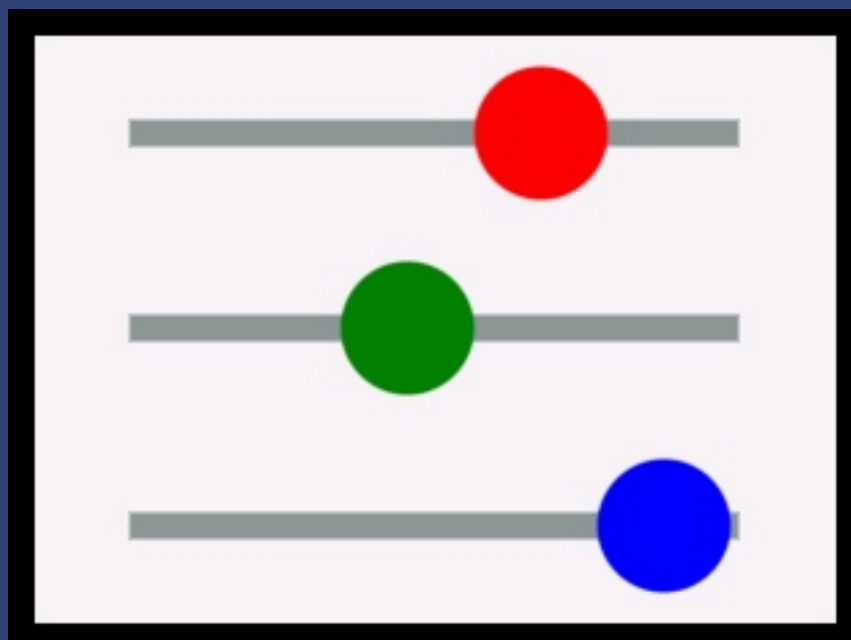


<color-palette>



ColorFactory

<rgb-slider>

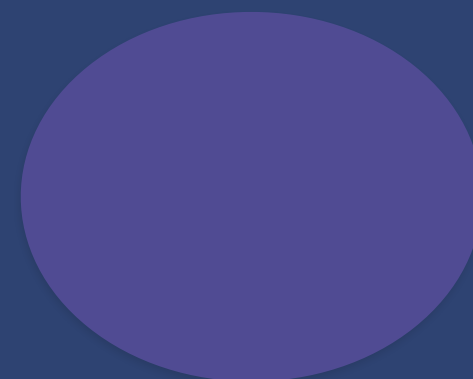
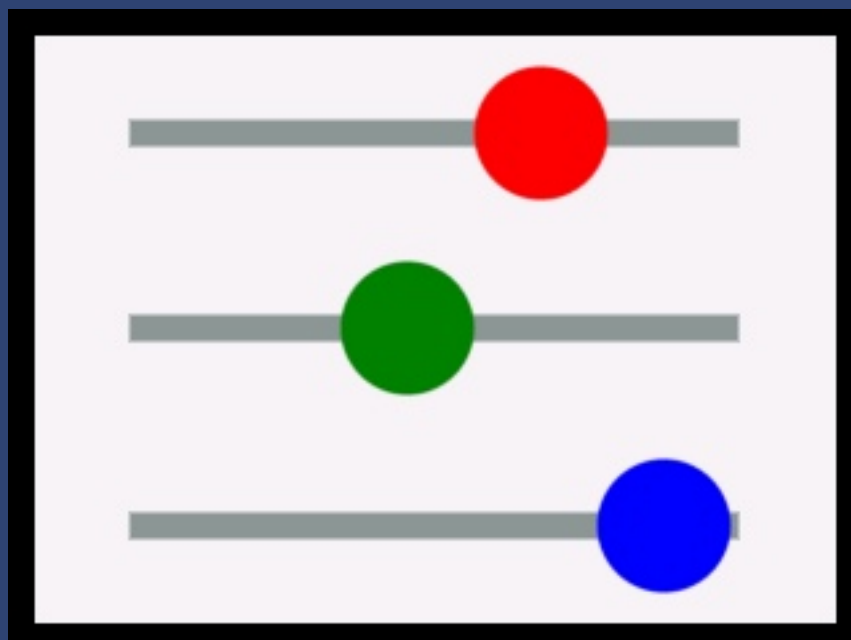


<color-palette>



ColorFactory

<rgb-slider>



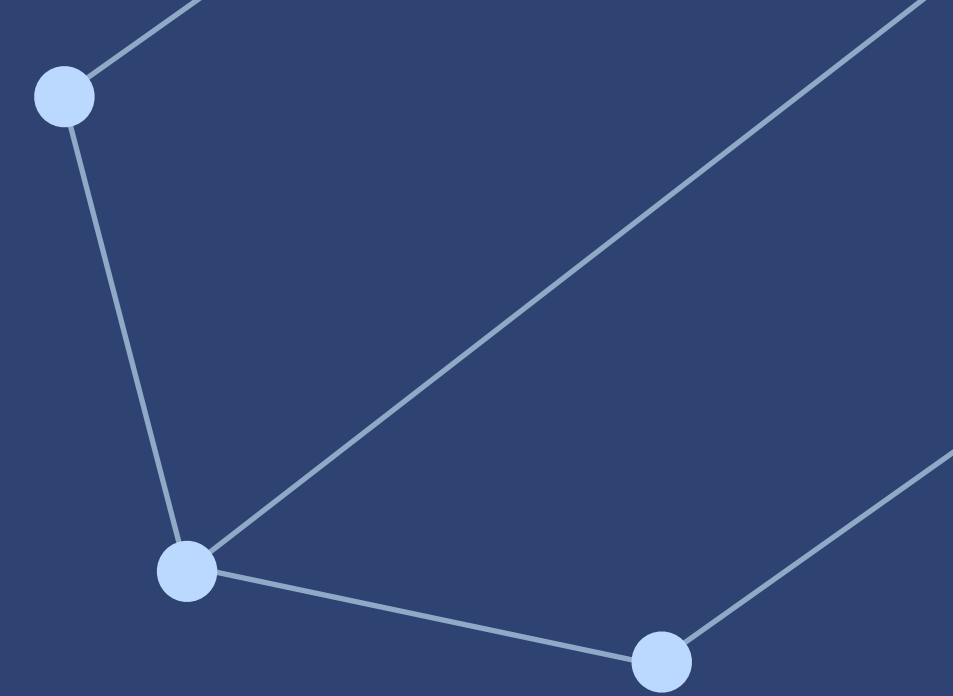
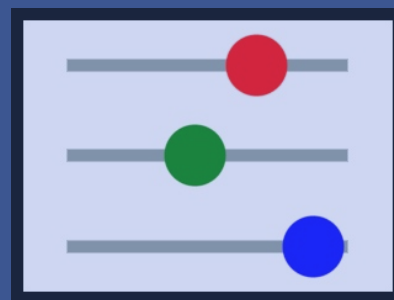
<color-palette>



<color-palette>



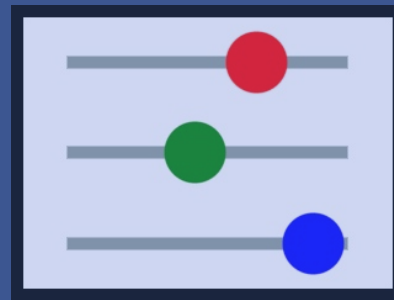
<rgb-slider>



<color-palette>



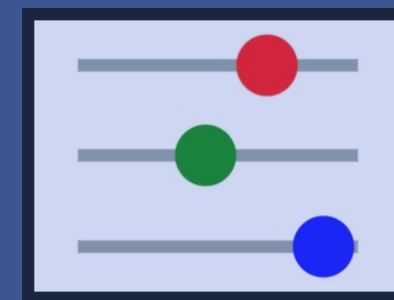
<rgb-slider>



<color-palette>



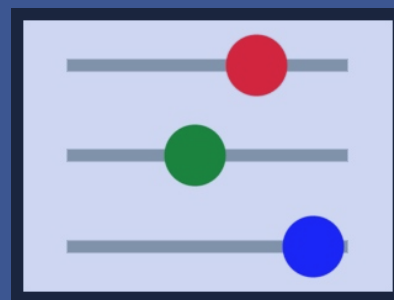
<rgb-slider>



<color-palette>



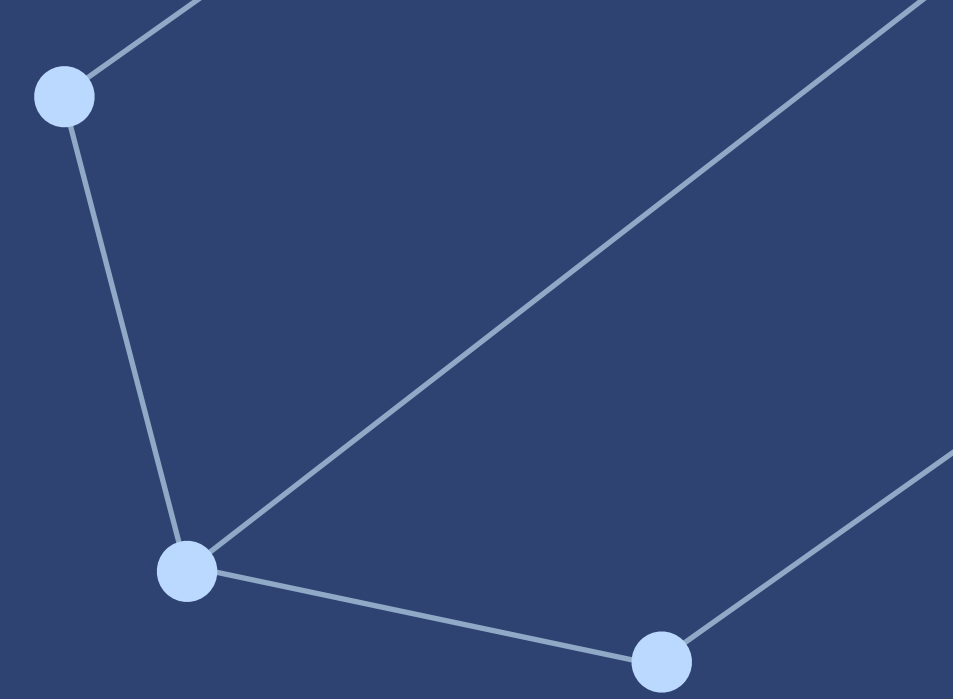
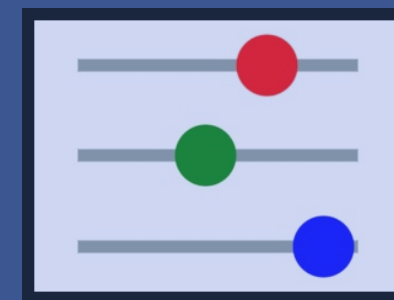
<rgb-slider>



<color-palette>



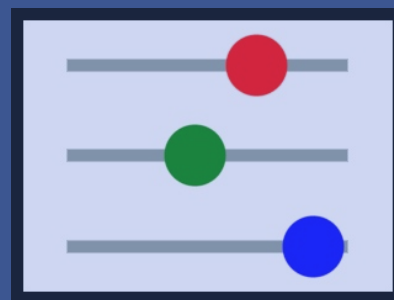
<rgb-slider>



<color-palette>



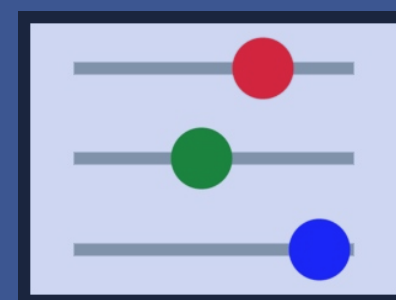
<rgb-slider>



<color-palette>



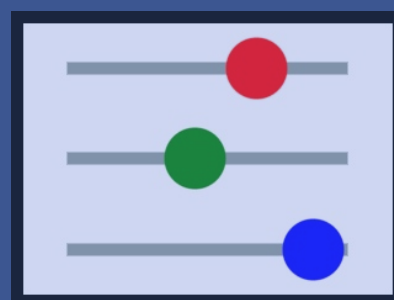
<rgb-slider>



<color-palette>



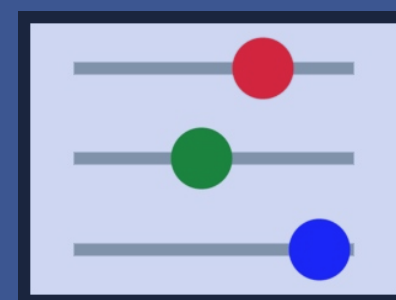
<rgb-slider>



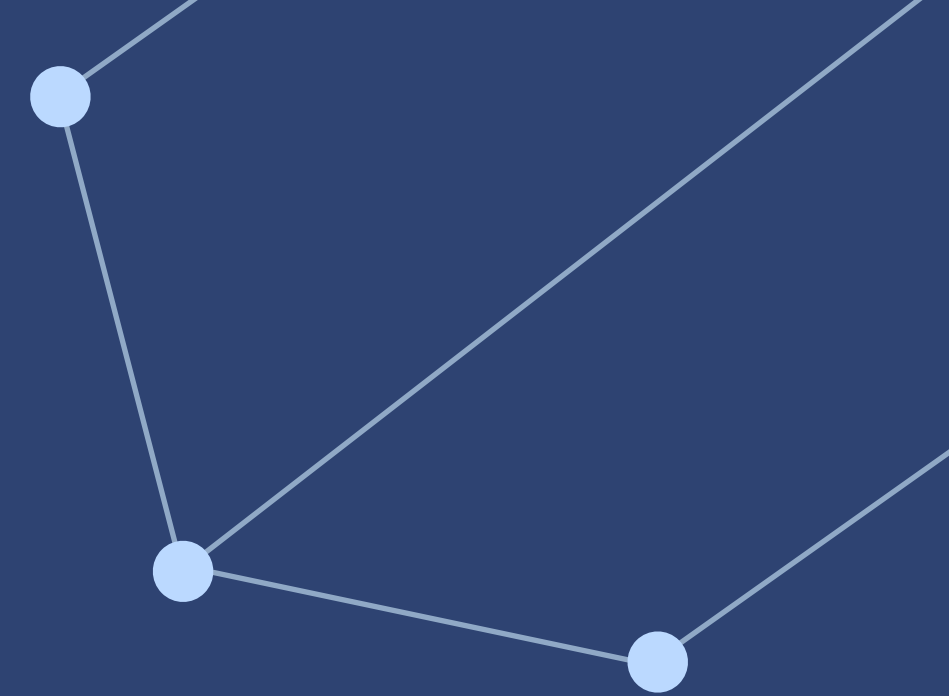
<color-palette>



<rgb-slider>



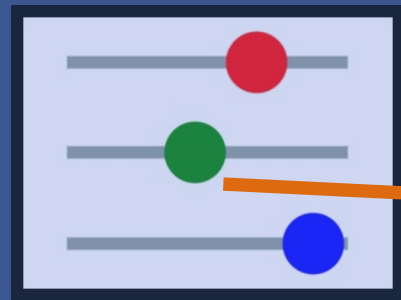
ColorFactory



<color-palette>



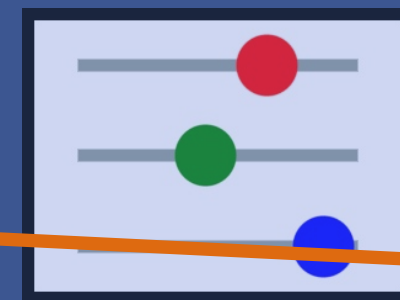
<rgb-slider>



<color-palette>



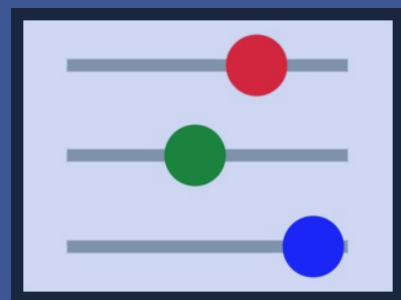
<rgb-slider>



<color-palette>



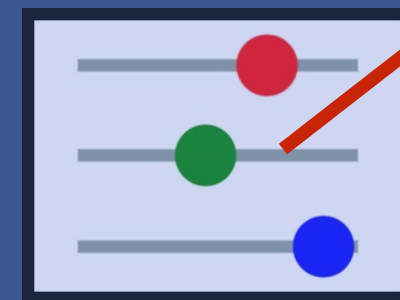
<rgb-slider>



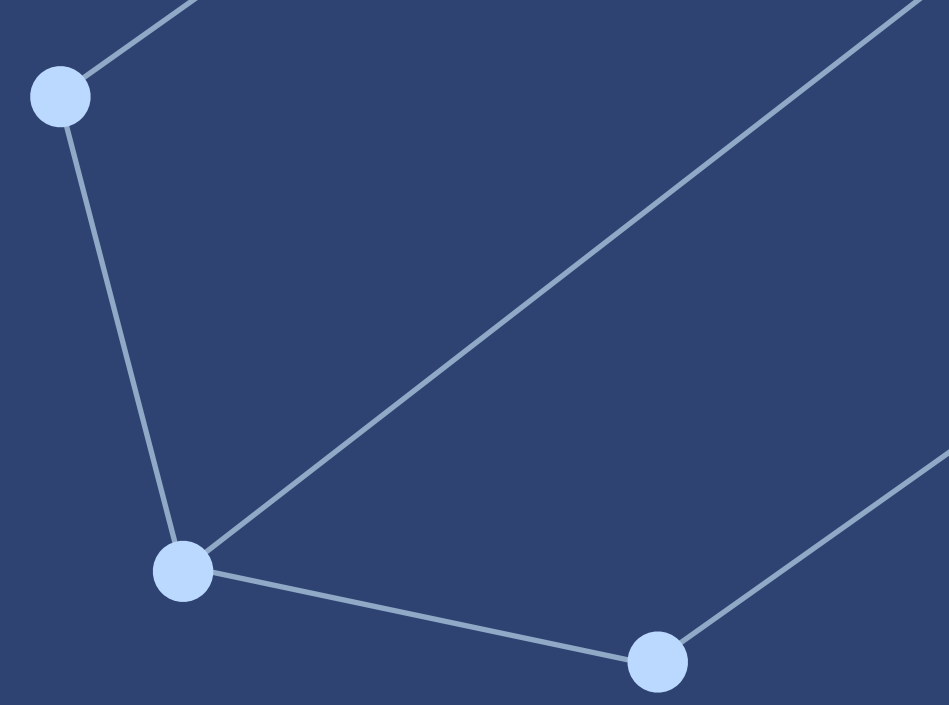
<color-palette>



<rgb-slider>



ColorFactory



require

```
<div id="example" first-directive second-directive></div>
```

```
app.directive('firstDirective', function () {  
    return {  
        restrict: 'A',  
        controller: function () {  
            this.firstDirectiveSays = function (message) {  
                console.log(message);  
            };  
        }  
    };  
});
```



```
app.directive('firstDirective', function () {  
    return {  
        restrict: 'A',  
        controller: function () {  
            this.firstDirectiveSays = function (message) {  
                console.log(message);  
            };  
        }  
    };  
});  
});
```

```
app.directive('secondDirective', function () {  
    return {  
        restrict: 'A',  
        require: 'firstDirective',  
        link: function (scope, element, attrs, firstDirCtrl) {  
            var m = 'Hello from second directive!';  
            firstDirCtrl.firstDirectiveSays(m);  
        }  
    };  
});
```

```
<div id="example" first-directive second-directive></div>
```

```
app.directive('secondDirective', function () {  
  return {  
    restrict: 'A',  
    require: 'firstDirective',  
    link: function (scope, element, attrs, firstDirCtrl) {  
      var m = 'Hello from second directive!';  
      firstDirCtrl.firstDirectiveSays(m);  
    }  
  };  
});
```

```
app.directive('secondDirective', function () {  
    return {  
        restrict: 'A',  
        require: 'firstDirective',  
        link: function (scope, element, attrs, firstDirCtrl) {  
            var m = 'Hello from second directive!';  
            firstDirCtrl.firstDirectiveSays(m);  
        }  
    };  
});
```

ngModelController

```
<input type="text" ng-model="search.text" />
```

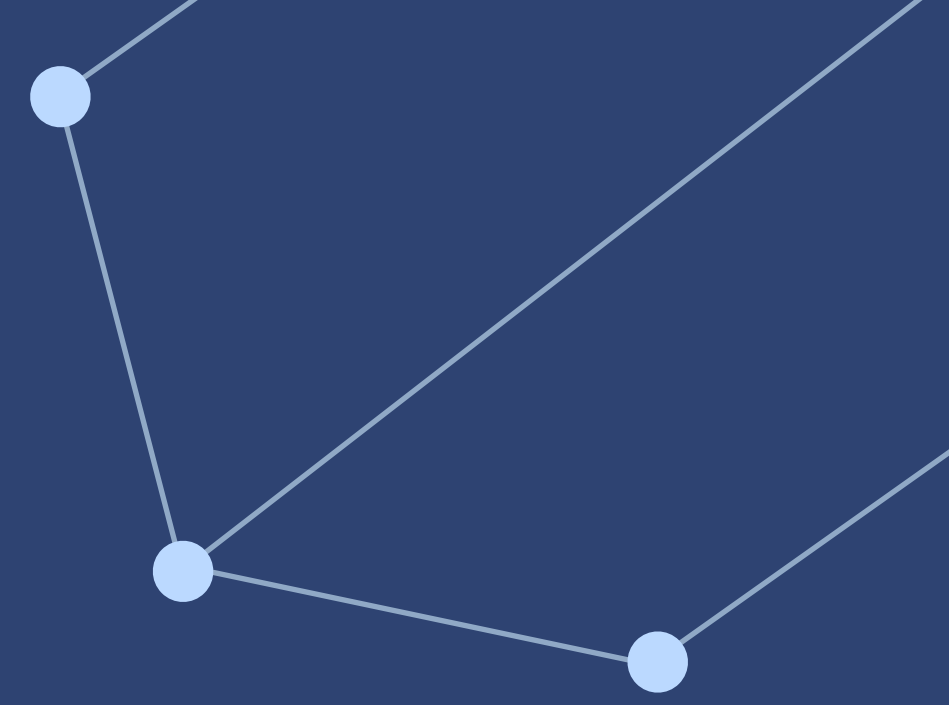
```
app.directive('myDirective', function () {  
    return {  
        require: 'ngModel',  
        link: function (scope, element, attrs, ngModelCtrl) {  
              
        }  
    };  
});
```



```
app.directive('myDirective', function () {  
  return {  
    require: 'ngModel',  
    link: function (scope, element, attrs, ngModelCtrl) {  
        
    }  
  };  
});
```

```
<input type="text" my-directive ng-model="search.text" />
```

Two-way Data Binding



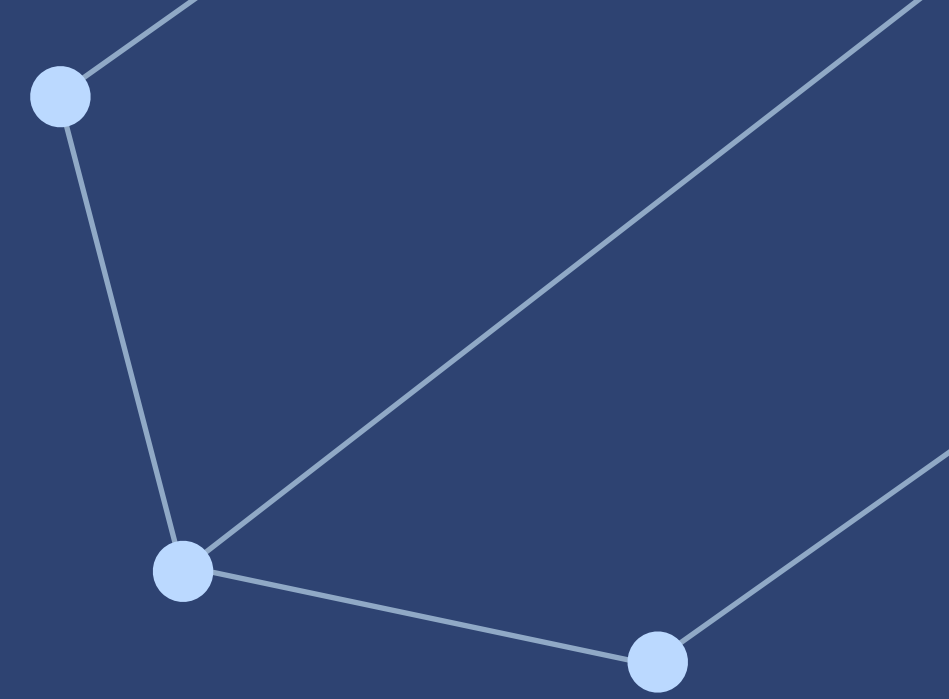


No input element works with
Angular by default.

ng-model

[\$viewValue]

Value as it appears in the input



ng-model

[\$viewValue]

Value as it appears in the input

```
<input type="checkbox" ng-model="rememberMe" />
```

ng-model

[\$viewValue]

Value as it appears in the input

```
<input type="checkbox" ng-model="rememberMe" />
```



inputDOM.checked

ng-model

[\$viewValue]

Value as it appears in the input

```
<input type="text" ng-model="username" />
```

ng-model

[\$viewValue]

Value as it appears in the input

```
<input type="text" ng-model="username" />
```

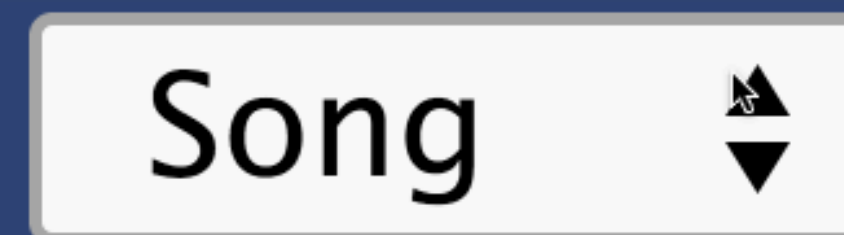
inputDOM.value

ng-model

[\$viewValue]

Value as it appears in the input

```
<select ng-model="sortBy">  
  <option>Song</option>  
  <option>Artist</option>  
  <option>Album</option>  
</select>
```

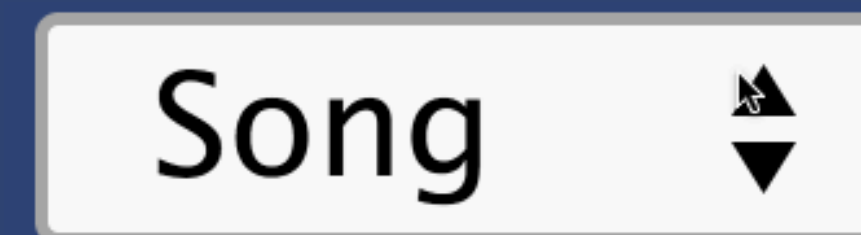
A UI representation of a dropdown menu. It consists of a light gray rounded rectangle containing the text 'Song' in a dark font. To the right of the text is a small black icon of a downward-pointing triangle, indicating that the menu can be expanded.

ng-model

[\$viewValue]

Value as it appears in the input

```
<select ng-model="sortBy">  
  <option>Song</option>  
  <option>Artist</option>  
  <option>Album</option>  
</select>
```

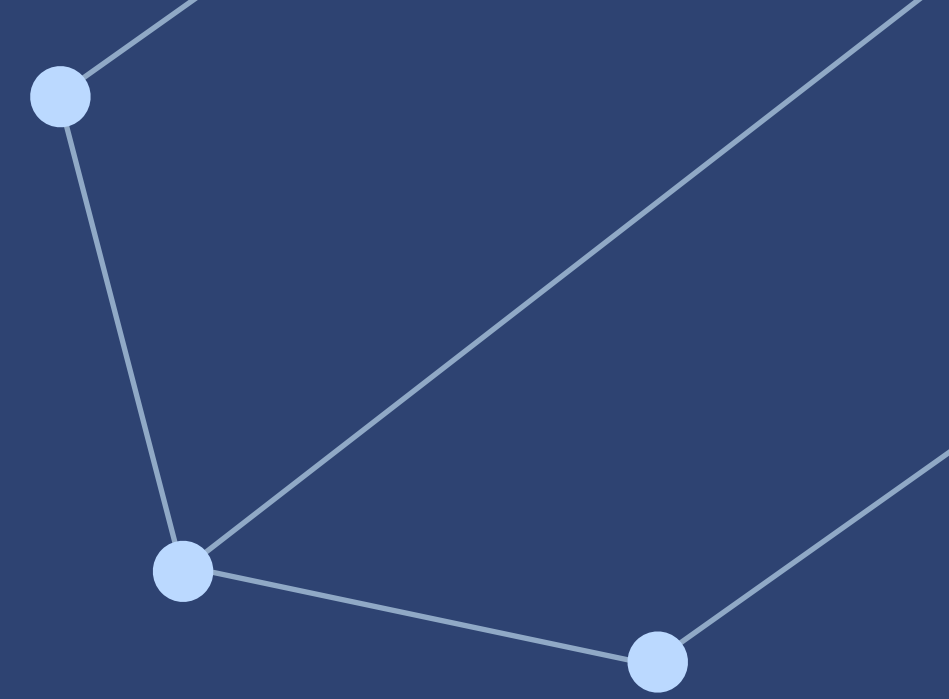
A white dropdown menu with a light gray border. The word "Song" is displayed in black text. To the right of the text is a small black icon consisting of two triangles pointing in opposite vertical directions.

```
selectDOM.options[selectDOM.selectedIndex].value
```

ng-model

[\$modelValue]

Value as it appears on the scope



ng-model

[\$modelValue]

Value as it appears on the scope

```
<input type="text" ng-model="username" />
```

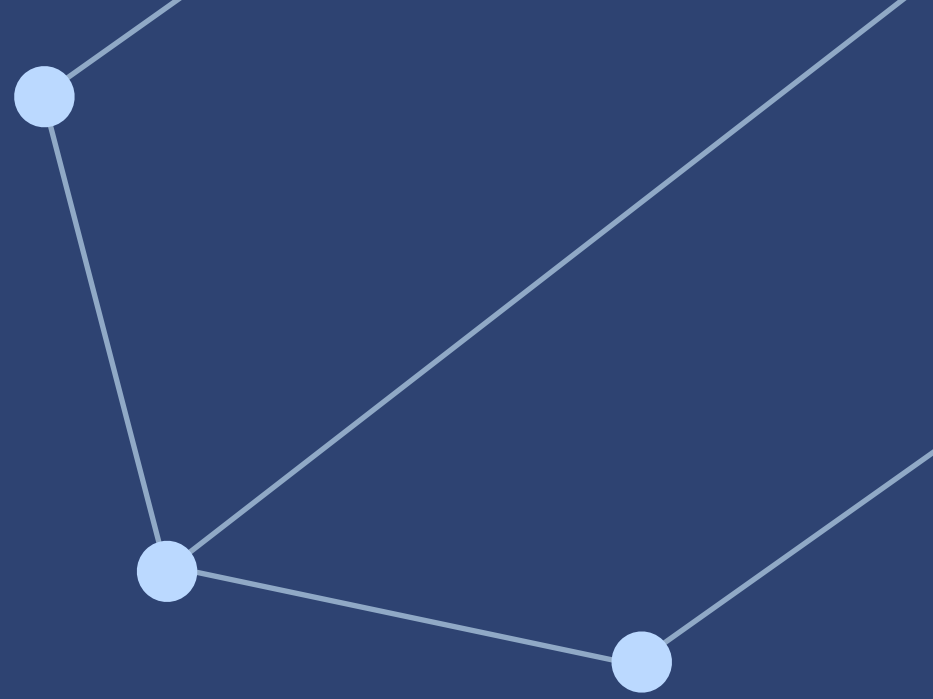
\$scope.username

\$setViewValue

[\$viewValue]



[\$modelValue]



\$setViewValue



1. Input changes in significant way



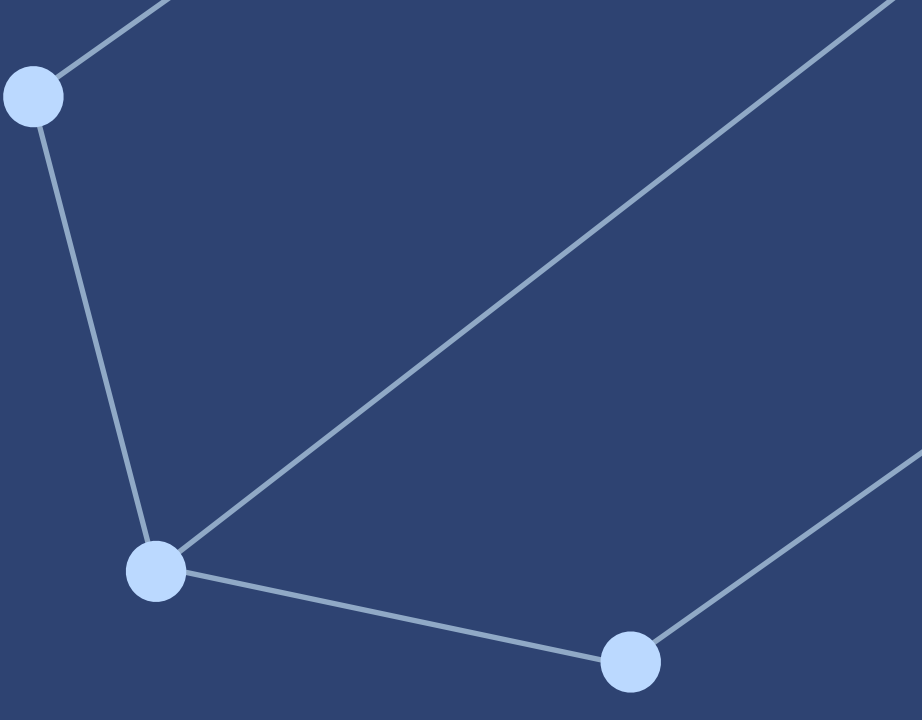
\$setViewValue



- 1.** Input changes in significant way
- 2.** `$setViewValue` called with updated value



```
ctrl.$setViewValue(element[0].checked, ev && ev.type);
```



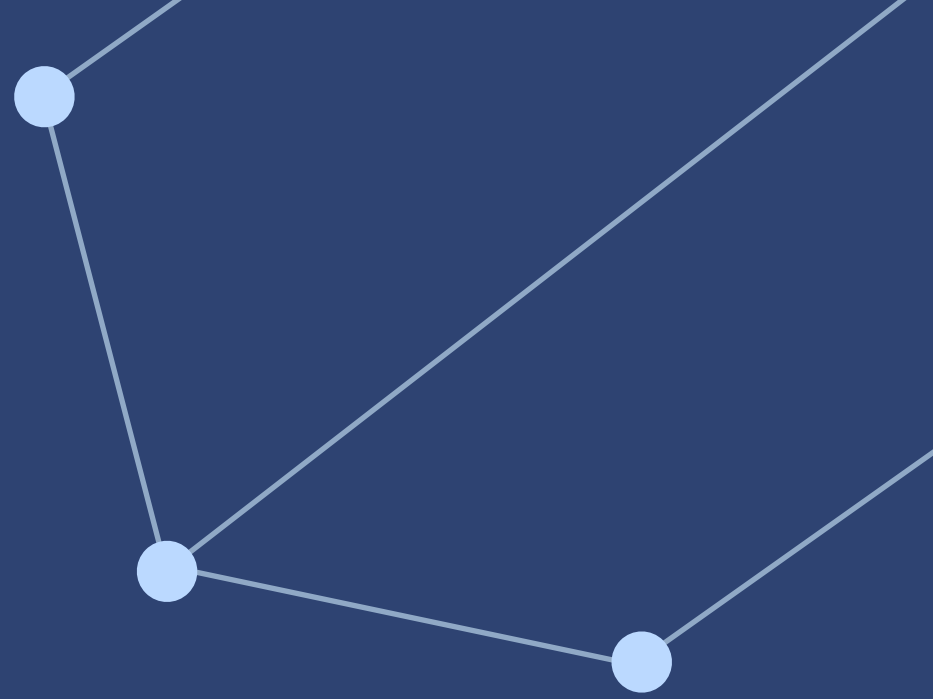
```
element.on('change', function() {  
    scope.$apply(function() {  
        ngModelCtrl.$setViewValue(selectCtrl.readValue());  
    });  
});
```

\$setViewValue

[\$viewValue]



[\$modelValue]



\$setViewValue

[\$viewValue]



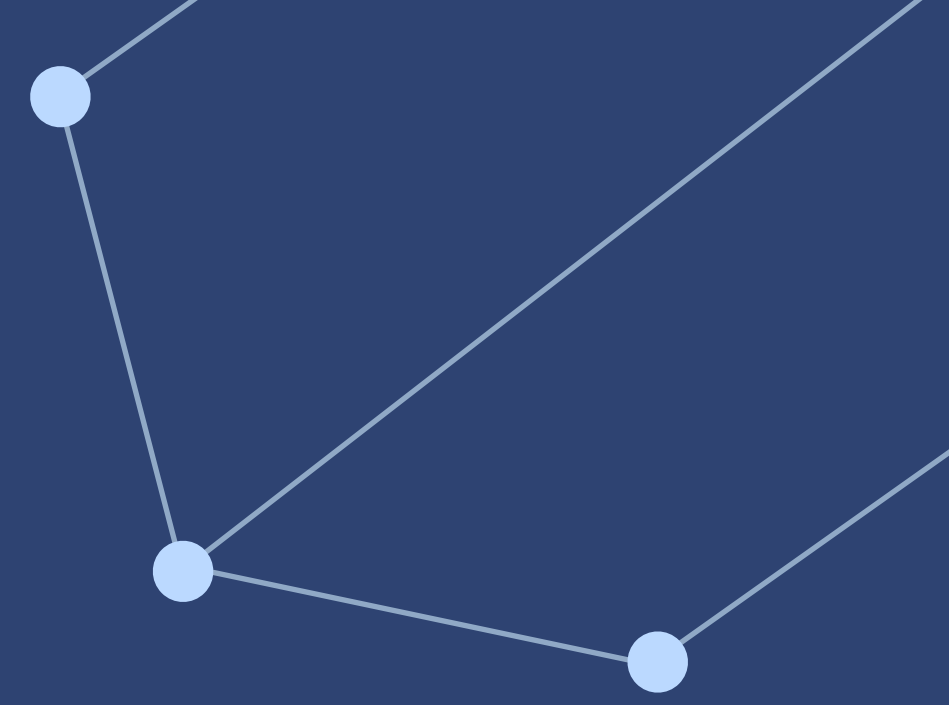
[\$modelValue]

[\$viewValue]



[\$modelValue]





[\$viewValue]



[\$modelValue]



1. `$watch` on `$scope` triggers change on `$modelValue`

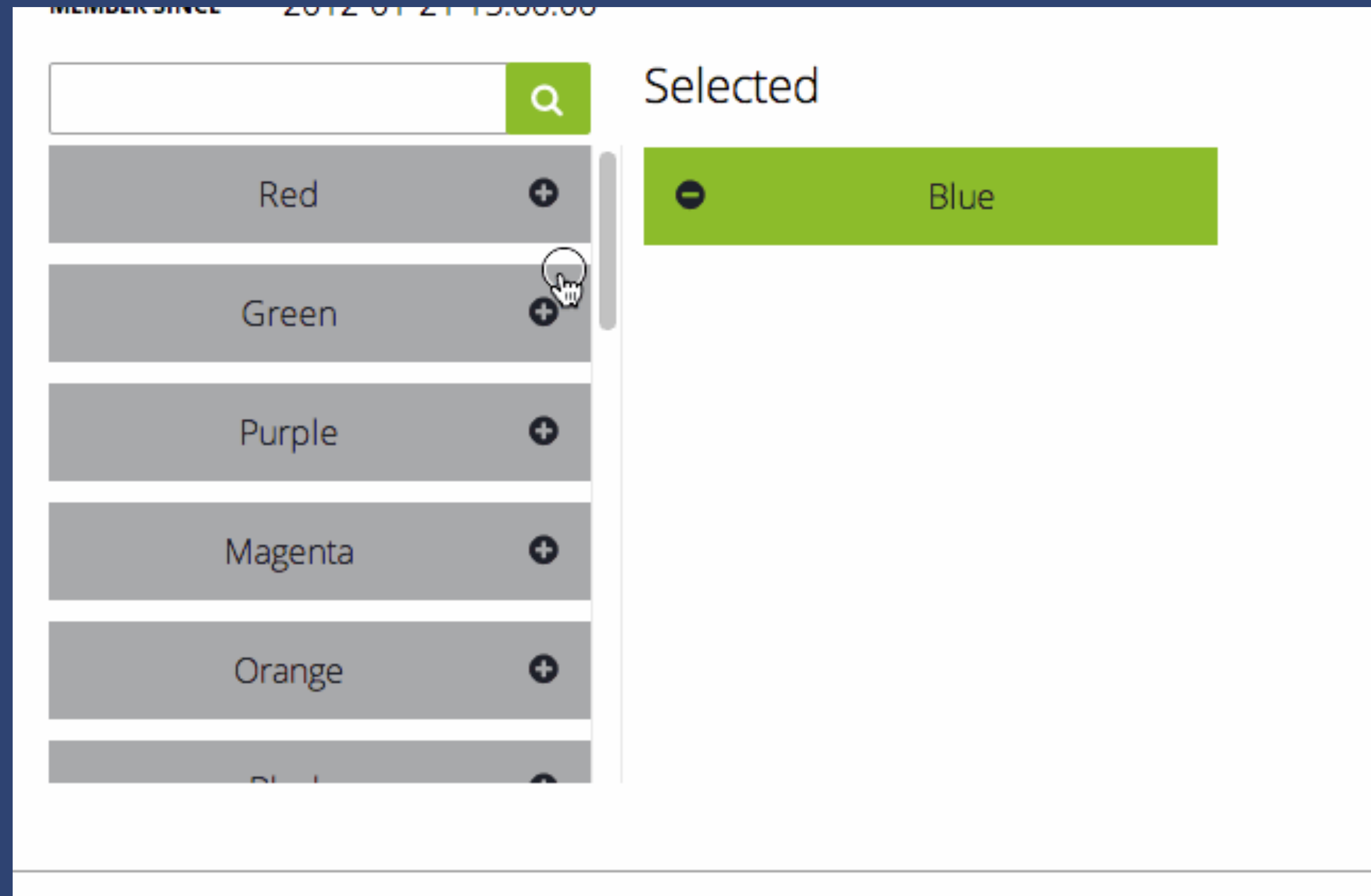


- 1.** `$watch` on `$scope` triggers change on `$modelValue`
- 2.** Reflect new value in input (`$render` function)

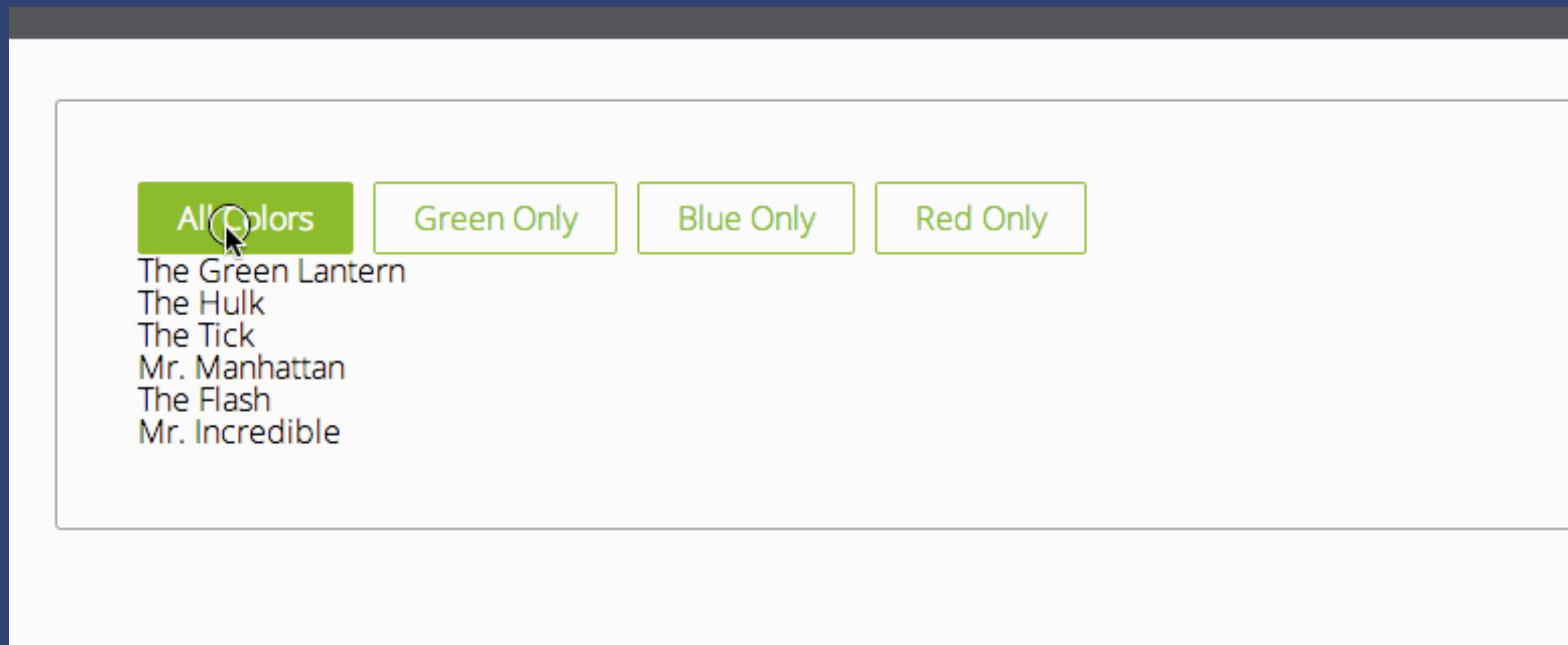

```
ctrl.$render = function() {  
    element[0].checked = ctrl.$viewValue;  
};
```

```
ctrl.$render = function() {  
    element.val(ctrl.$isEmpty(ctrl.$viewValue) ? '' : ctrl.$viewValue);  
};
```

```
ngModelCtrl.$render = function() {  
    selectCtrl.writeValue(ngModelCtrl.$viewValue);  
};
```



```
<adjacent-lists  
    ng-model="chosenColors"  
    items="colors">  
</adjacent-lists>
```



```
<filter-buttons  
    ng-model="chosenFilter"  
    items="choices">  
</filter-buttons>
```

Thanks.

