var app = angular.module(“myApp”, []); ~~ in .js file ~~

- This creates a new module named myApp.
- A module contains the different components of an AngularJS app.
- angular.module(name, [requires], [configFn]);
    - name – type: string; The name of the module to create or retrieve.
    - requires (optional) – type: !Array.<string>=; If specified then new module is being created. If unspecified then the module is being retrieved for further configuration.
    - configFn(optional) – type: Function=; Optional configuration function for the module.

<body ng-app=”myApp”> ~~ in .html file ~~

- The ng-app is called a directive. It tells AngularJS that the myApp module will live within the <body> element, termed the application’s scope. In other words, we used the ng-app directive to define the application scope.

```
app.controller('MainController', ['$scope', function($scope) {
  $scope.title = 'Top Sellers in Books';
}]);
```
~~ in MainController.js file ~~

- Creates a new controller named MainController. A controller manages the app’s data. Here we use the property title to store a string, and attach it to $scope.

 ~~ in .html file ~~

- ng-controller is a directive that defines the controller scope. This means that properties attached to $scope in MainController because available to use within

{{ title }}

- This is an expression and is used to display values on the page. In , this is how we would access $scope.title.

```
$scope.product = {
                    name: ‘The Book of Trees’,
                    price: 19
                }
```

- Makes use of an object to group together related data about a product.

 {{ product.name }}

 {{ product.price }}

- Use dot notation to display the values.

{{ product.price | currency }}

- AngularJS gets the value of product.price and pipes this number into the currency filter.

pubdate: new Date('2014', '03', '08')

 {{ product.pubdate | date }}

- Another example of an AngularJS filter.

Here's a quick review!

- A module contains the different components of an AngularJS app.
- A controller manages the app's data.
- An expression displays values on the page.
- A filter formats the value of an expression.

```
$scope.products = [
                    {
                    name: 'The Book of Trees',
                    price: 19,
                    pubdate: new Date('2014', '03', '08'),
                    cover: 'img/the-book-of-trees.jpg'
                    },
                    {
                    name: 'Program or be Programmed',
                    price: 8,
                    pubdate: new Date('2013', '08', '01'),
                    cover: 'img/program-or-be-programmed.jpg'
                    }
                ]
```

- This is how you make an array of objects.
- `<div ng-repeat="product in products" class="col-md-6" >`
    - Adding this in the view uses the ng-repeat directive to loop through an array and display each element. It repeats all the HTML inside for each element in products array.
    -

```
$scope.plusOne = function(index) {
        $scope.products[index].likes += 1;
};
```

- How to attach a function to scope in the controller.

```
<div class="rating">
  <p class="likes" ng-click="plusOne($index)">+ {{ product.likes }}</p>
</div>
```

- The ng-click is a directive that tells AngularJS to run the plusOne() function in the controller.

Generalization of what we've covered so far

1. A user visits the AngularJS app.
2. The view presents the app's data through the use of expressions, filters, and directives. Directives bind new behavior HTML elements.
3. A user clicks an element in the view. If the element has a directive, AngularJS runs the function.
4. The function in the controller updates the state of the data.
5. The view automatically changes and displays the updated data. The page doesn't need to reload at any point.