ANGULARJS

WELCOME BACK!

GOALS FOR THIS UNIT

1. Review
2. Intro to Angular
3. Single Page Application & MVC
4. Directives & Data binding
5. Filters
6. Controllers
    - Scope
    - Dependency Injection

QUESTIONS?

REVIEW

INTRO TO ANGULAR

WHAT IS ANGULAR?

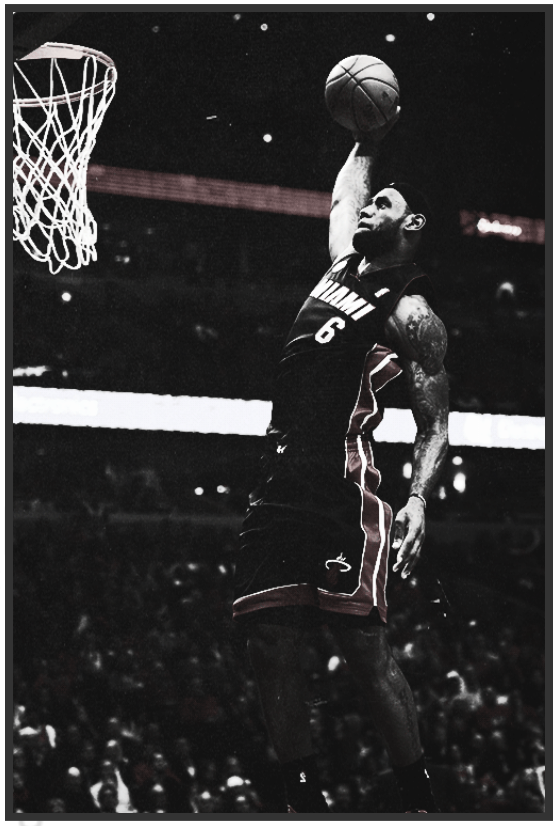
Angular is a front-end MVC JavaScript framework intended to simplify making robust web applications. The overall ethos of the Angular project is 'What if HTML had been developed today from scratch?'. To that end, much of Angular's functionality is geared toward extending HTML's natural capabilities to make it better suited to support modern web applications.

SINGLE PAGE APPLICATIONS (SPA)

SINGLE PAGE APPLICATIONS

A relatively recent trend in web design, single page apps have become the standard as opposed to the exception. In general, a SPA can be characterized by having an outer 'shell' that serves as the header and navigation for the site while the content of the page changes as different parts of the site are visited.

MODEL VIEW CONTROLLER (MVC)


MIAMI
6

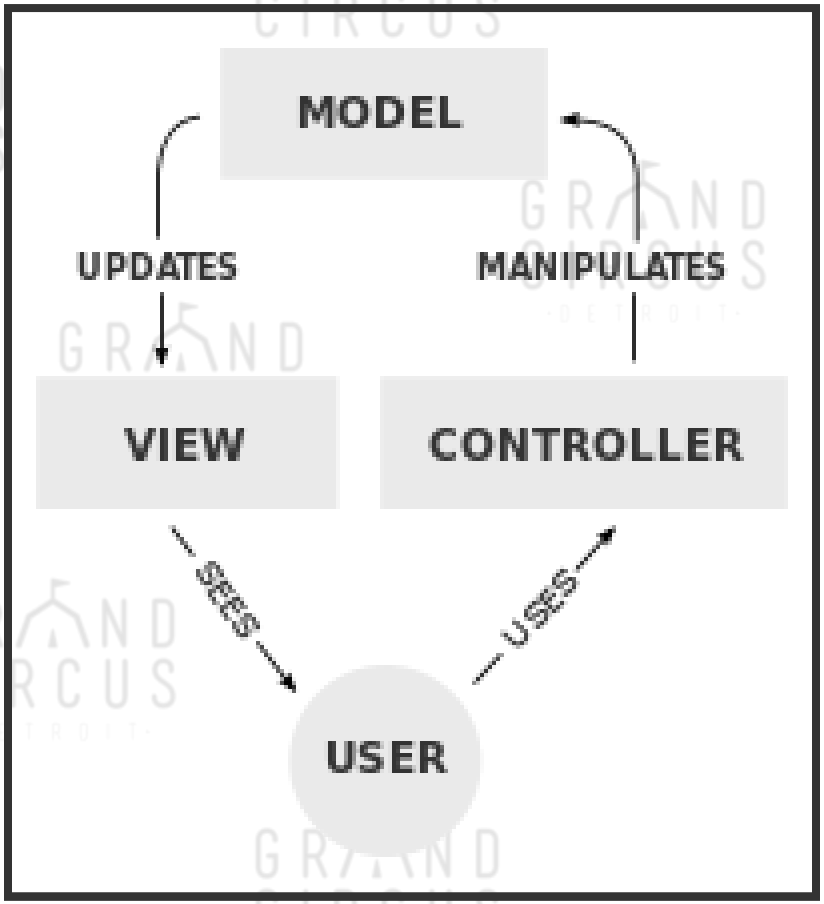
NO, NOT MVP!

MVC is a design pattern that has become ubiquitous in the last few years. It describes the relationship between the various parts of a modern web app. MVC has evolved into a number of other sub-types (Model View View-Model, Model View Presenter, Model View *) but the basic concept is the same across each of these evolutions.

THE COMPONENTS

- Model: the data
- View: what the user sees
- Controller: the logic that brings it all together

MVC



GETTING STARTED

DOWNLOAD

http://angularjs.org

Angular can also be referenced remotely via a number of CDNs.

IMPORTING THE SCRIPT

Pop this in at the end of the body of your HTML.

```
<script src="lib/angular.js"></script>
<!-- OR -->
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.min.js"></
```

DIRECTIVES

DIRECTIVES

Directives are Angular's way of extending native HTML by creating custom HTML elements attributes.

ANGULAR DIRECTIVES

The canonical first Angular example

```
<div ng-app="">
  <input type="text" ng-model="things" class="ng-pristine ng-untouched ng-valid">
  <h1 class="ng-binding">{{things}}</h1> <!-- Angular expression -->
</div>
```

DEMO

DIRECTIVES

Directive	Description
`ngApp`	Declares an element and all its children as an angular app
`ngModel`	Binds a form control to a property on the scope*

* - More on this later

NORMALIZED VS. DENORMALIZED

You'll notice that we refer to Angular directives in a couple of different ways. You can check the Angular docs for a more detailed explanation, but this is largely because HTML is case-insensitive. We'll use the denormalized form to refer to directives in the DOM.

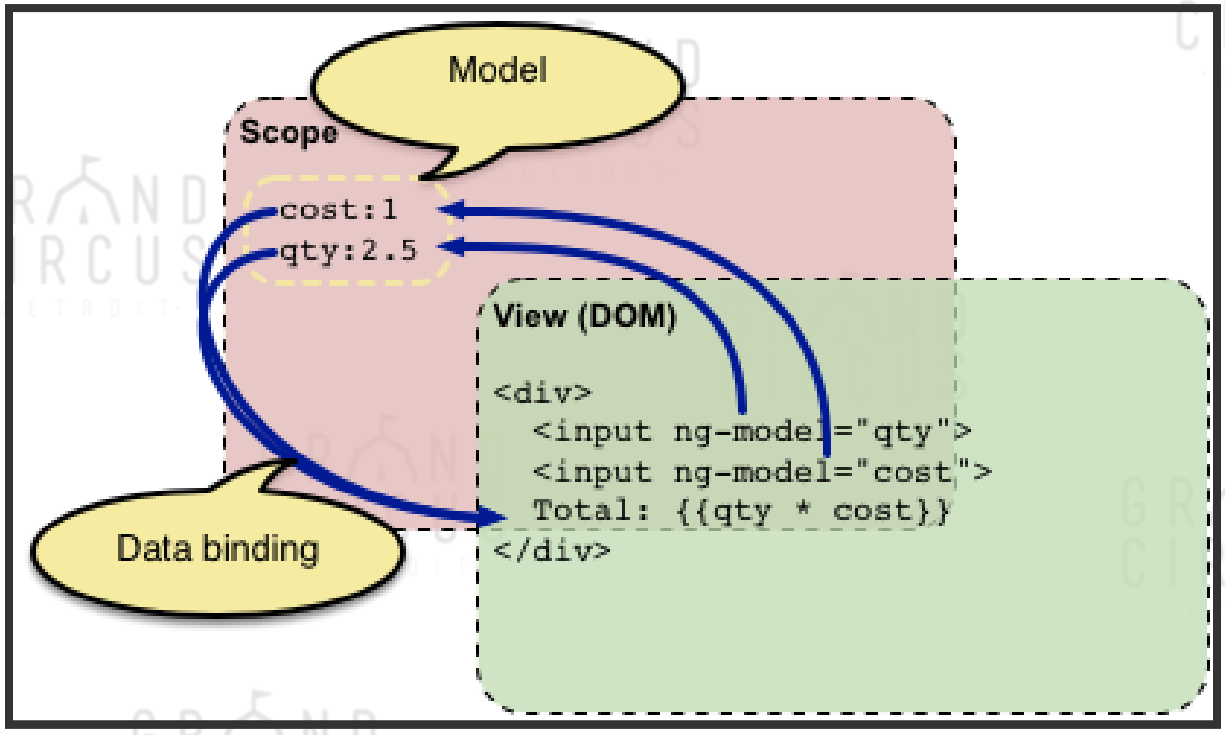
- Normalized: case-sensitive, camelCase (e.g. ngApp)
- Denormalized: lower-case, dash-delimited (e.g. ng-model)

DATA BINDING

DATA BINDING

Data binding is one of the key features that make Angular so powerful. You can set certain properties and tell Angular to 'watch' those properties. Whenever those properties change during program execution, the view updates them on the view automagically. We saw this at work in our very first Angular demo.

DATA BINDING



ANGULAR EXPRESSIONS

ANGULAR EXPRESSIONS

The double curly braces `{{` `}}` from the example are Angular's way of knowing what it needs to evaluate. This is a relatively deep well, but suffice it to say that *one* of the things an expression will do is scan the scope (more later, I swear) for any variables of that name and do something with it.

MATH

We can evaluate mathematical expressions as well.

```
<span>
  1+2 = {{1+2}}
</span>
```

or string operations

```
<span ng-init="mood='happy'">
  My mood is {{ mood + "!" }}
</span>
<!-- I'm aware we haven't talked about ng-init yet. Don't freak out we will. -->
```

LOGICAL OPERATORS

We can evaluate logical operators.

```
<span>
true || false  :  {{ true || false}}
<br>
false || false  : {{ false || false  }}
<br>
true && false  : {{ true && false  }}
<br>
!true  : {{ !true  }}
</span>
```

EXERCISE!

DATA BINDING

POOR MAN'S MAD-LIBS™

Code with me!

1. Set up a new project. (index.html)
2. Add Angular to your project. (download or CDN)
3. Add an `ngApp` directive to your app.
4. Add two text input fields to your page.
5. Add an `ngModel` directive to each input tag, giving them values of "noun" and "adjective".
6. Add a heading tag that uses two Angular expressions, one for each model.
7. Compose a simple sentence that allows the user to add words to the sentence by filling out the text inputs.

EXAMPLE

EXERCISE!

MORE DATA BINDING

MAKE A SIMPLE ANGULAR APP

Demonstrate a basic understanding of Angular data binding.

- Create another new project and include Angular
- Add several variations of our Mad-Lib example from the previous example (Data bound elements)
- At least 3 different usages of Angular's expressions:
  - Math
  - String operations
  - Logical operators

MOAR DIRECTIVES

DIRECTIVES

`ngRepeat` is maybe one of the most awesome things about Angular.

```
<div ng-init="beatles=['John', 'Paul', 'George', 'Ringo']">

<h3>Names of The Beatles</h3>
  <ul>
    <li ng-repeat="beatle in beatles">{{beatle}}</li>
  </ul>
</div>
```

DEMO

MORE ANGULAR DIRECTIVES

Directive	Description
`ngRepeat`	The ngRepeat directive instantiates a template once per item from a collection.
`ngInit`	The ngInit directive allows you to evaluate an expression in the current scope.

EXERCISE!

DIRECTIVES

FAVORITE BAND ROSTER

List the members of your favorite band using `ngRepeat`.

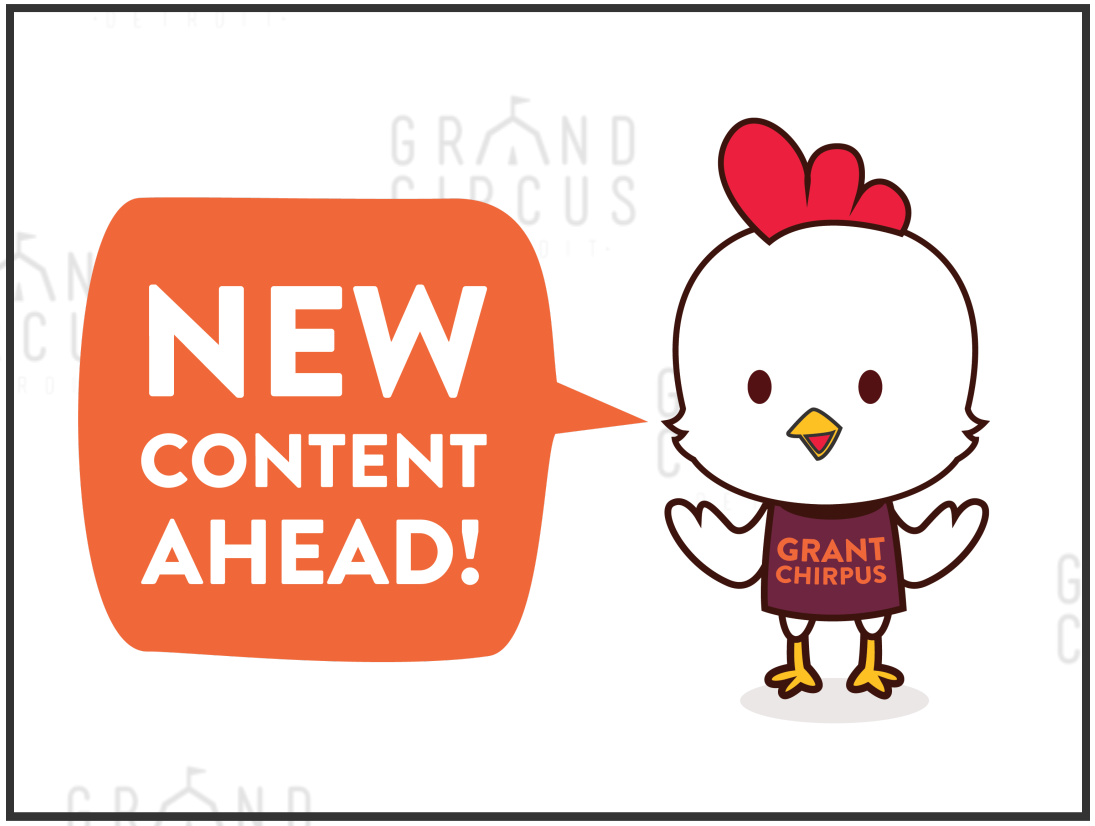
- Set up a new project (index.html) and add Angular.
- Add an `ng-app` directive to your app.
- Add a container element with an `ngInit` with the names of your favorite band as an array of strings.
- Use `ngRepeat` to repeat a template for each item in the array.

BONUS!

- Use something other than list items for your template. You can use `ngRepeat` to iterate anything.

Be ready to demo!

DEMOS


NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

FILTERS

FILTERS

Filters format the value of an expression for display to the user. Filters can also be used to do in-client searches to... well... *filter* a data set.

FILTER USAGE

Use the pipe operator in an Angular expression to use a filter on it.

```
<div ng-init="theWho=['Roger Daltry', 'Pete Townsend', 'Keith Moon']">
  <h3>Members of The Who in uppercase</h3>
  <ul>
    <li ng-repeat="member in theWho">{{ member | uppercase }}</li>
  </ul>
</div>
```

DEMO

MORE FILTERS

Filter	Description
`uppercase`	Uppercases the output
`lowercase`	Lowercases the output
`orderBy`	will reorder the data based on a pre-determined property
`filter`	allows a data set to be fuzzy searched
`limitTo`	will limit the iterations to a specified number

ORDERBY

DEMO

```
<div ng-init="theWho=['Roger Daltry', 'Pete Townsend', 'Keith Moon']">
  <h3>Members of The Who in alphabetical order</h3>
  <ul>
    <li ng-repeat="member in theWho | orderBy">{{ member }}</li>
  </ul>
</div>
```

```
<h3>Members of The Who in reverse alphabetical order</h3>
<ul>
  <li ng-repeat="member in theWho | orderBy:'-'">{{ member }}</li>
</ul>
```

FILTER FILTER

This example is too big for a slide. Let's look at the demo.

DEMO

THINGS TO NOTICE:

- The item intialized in the `ng-init` is an array of objects.
- We can access the individual object properties in the array using dot notation.
- We're using `ng-model` to capture the search term.
- The search is fuzzy. It filters on name and instrument.

CONTROLLERS

CONTROLLERS

Controllers handle the business logic behind views. These are the primary means of controlling the UI. Their primary role is to expose variables and functionality to expressions and directives.

CONTROLLERS

Our Grateful Dead example using controllers instead of `ngInit`.

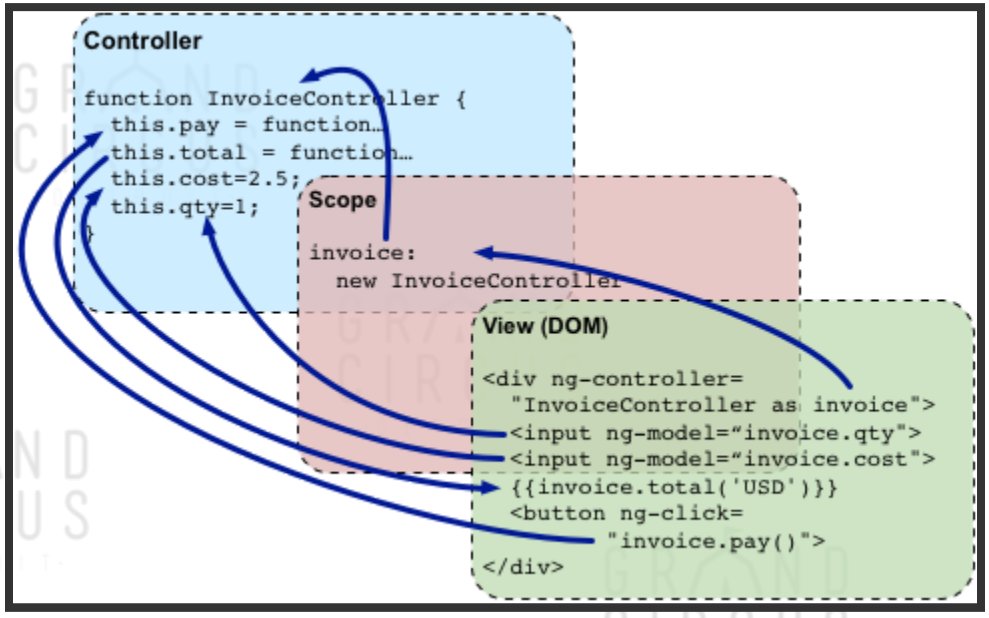
Again, this example is too big for the slide.

DEMO

SCOPE

SCOPE

Scope is Angular's 'glue object' that marries the variables and properties on a controller to the view.



SCOPE

In order to take advantage of the Scope object we must inject it.

```
app.controller('simpleController', function($scope) {
  $scope.theDead = [
    {name: 'Jerry Garcia', instrument: 'guitar, vocals'},
    {name: 'Bob Weir', instrument: 'guitar, vocals'},
    {name: 'Ron \'Pigpen\' McKernan', instrument: 'keyboards, harmonica, vocals'},
    {name: 'Phil Lesh', instrument: 'bass, vocals'},
    {name: 'Bill Kreutzmann', instrument: 'drums'}
  ];

});
```

This is Angular's version of Dependency Injection.

DEPENDENCY INJECTION

GROSS OVER-SIMPLIFICATION INBOUND

DEPENDENCY INJECTION

Dependency Injection is a concept in software design that allows for the components of a software project to be loosely coupled. This makes them easier to test and change without affecting the other modules that depend on them.

DEPENDENCY INJECTION

In Angular, software components (modules, services, and directives) are injected by passing them into the constructor function of whatever it is you're instantiating.

Note: This is the conventional way to define a controller.

```
var app = angular.module('myModule', []);
// This is declaring a module. More on this in a moment

app.controller('myController', function($scope){
  // controller logic
});
```

THINGS TO NOTICE

- First we create a module, which we then attach our controller to. We will go into more detail about modules next.
- We register a controller with our new module and give it a constructor function. This function will be run whenever a view that uses this controller is loaded.

CONTROLLER EXAMPLE

DEMO

```
<div ng-app="myModule" ng-controller="myController">
  <ul>
      <li ng-repeat="beatle in beatles">{{beatle}}</li>
  </ul>
</div>
```

```
var app = angular.module('myModule', []);
// This is declaring a module. More on this in a moment

app.controller('myController', function($scope){
  $scope.beatles = ['John', 'Paul', 'George', 'Ringo'];
});
```

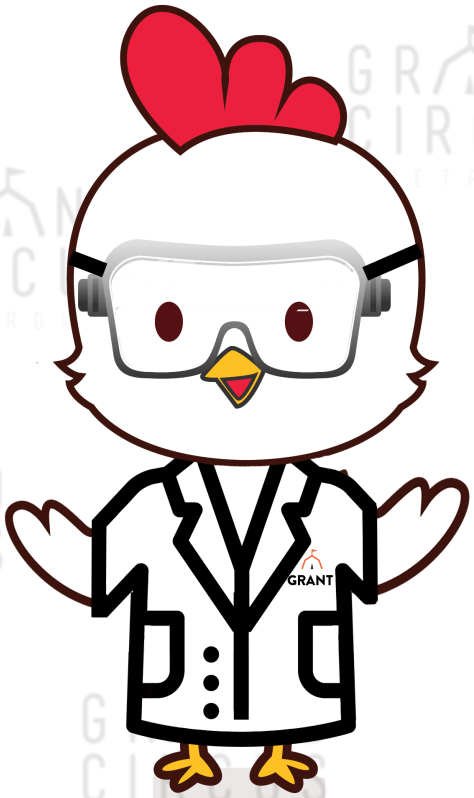
HOMEWORK

From ng-book:

- Modules: 19-20
- Angular Module Loading: 135-138

LAB 12

TO-DO WITH ANGULAR



INSTRUCTIONS

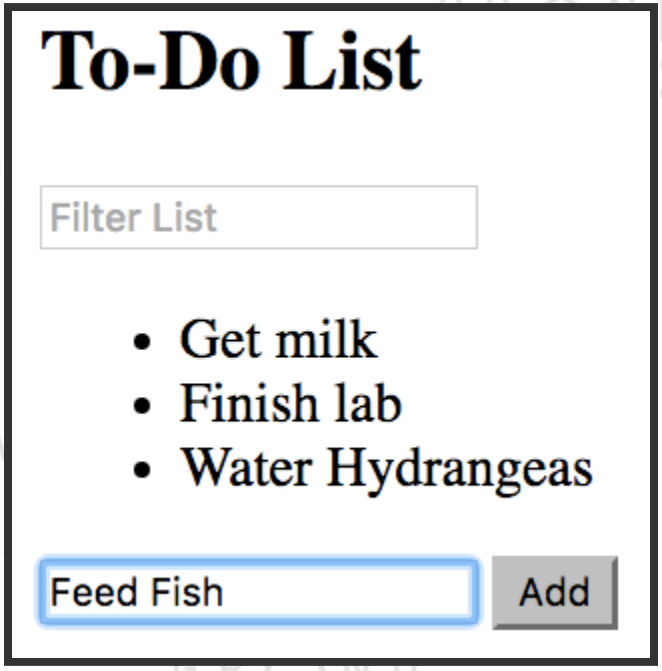
Create a to-do list app using Angular directives, filters and controllers.

- Create an HTML page with an input field and an "add" button.
- Create an array of strings to seed the list. Use `ng-repeat` to display them.
- When the user puts a to-do in the field and clicks add:
  - The info from the input field is added to the list.
- List should be filterable.

NOTES

- You'll need to make a module. Use the line from the previous examples.
- I have intentionally not given you everything you need to complete this lab. Have fun figuring it out, padawan learners.

PRE-GAME


To-Do List
Filter List
Get milk
Finish lab
Water Hydrangeas
Feed Fish
Add