GRAND CIRCUS

·DETROIT·

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?

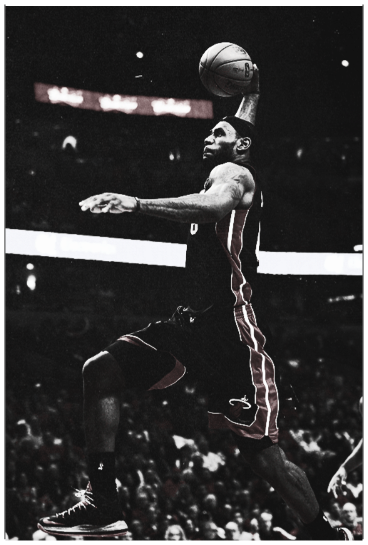
Angular is a front-end MVC JavaScript framework intended to simplify making robust web applications. The overall ethos of the Angular project is 'What if HTML had been developed today from scratch?'. To that end, much of Angular's functionality is geared toward extending HTML's natural capabilities to 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 o the page changes as different parts of the site are visited.

MODEL VIEW CONTROLLER (MVC)



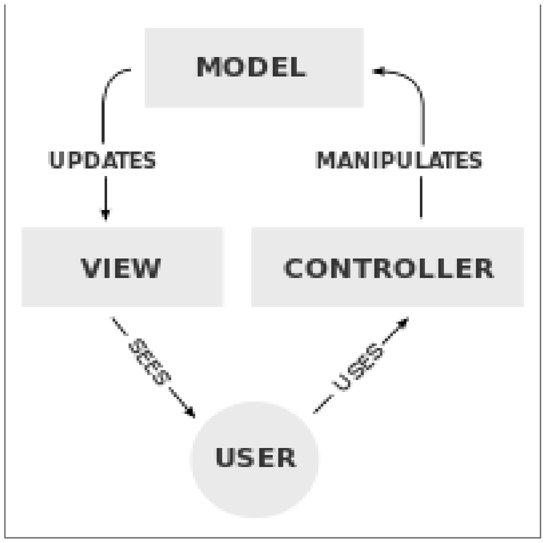
NO, NOT MVP!

MVC is a design pattern that has become ubiquitous in the last few years. It describe 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.

```
ib/angular.js"></script>
```

DIRECTIVES

DIRECTIVES

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

ANGULAR DIRECTIVES

The canonical first Angular example

```
>
"text" ng-model="things">
}}</h1> <!-- Angular expression -->
```

DEMO

DIRECTIVES

Directive	Description
	Declares an element and all its children as an angular app
	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 ca check the **Angular docs** for a more detailed explanation, but this is largely becaus HTML is case-insensitive. We'll use the denormalized form to refer to directives in the DOM.

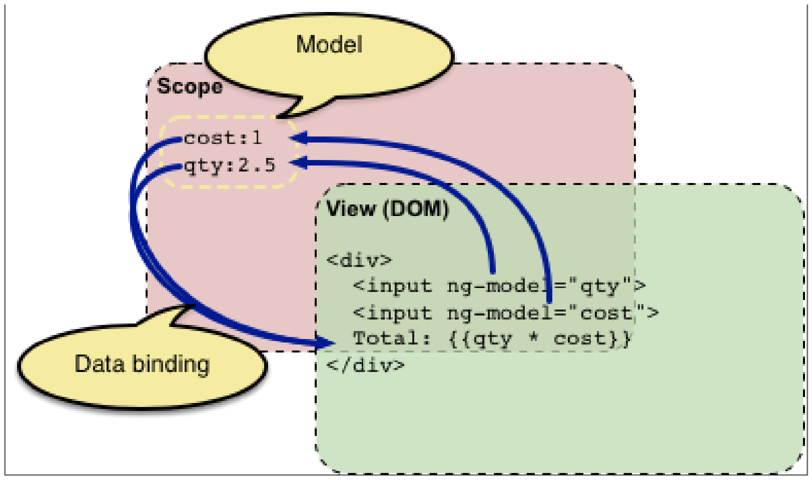
- Normalized: case-sensitive, camelCase (e.g. ngApp)
- Denormalized: lower-case, dash-delimited (e.g. ng-model)

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



```
<div>
  <input ng-model="qty">
  <input ng-model="cost">
  Total: {{qty * cost}}
</div>
```

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.

```
}
```

or string operations

```
"mood='happy'">
{ mood + "!" }}
```

```
 we haven't talked about ng-init yet. Don't freak out we will. -
```

LOGICAL OPERATORS

We can evaluate logical operators.

```
 :  {{ true || false}}
  : {{ false || false  }}
 : {{ true && false  }}
rue  }}
```

EXERCISE!

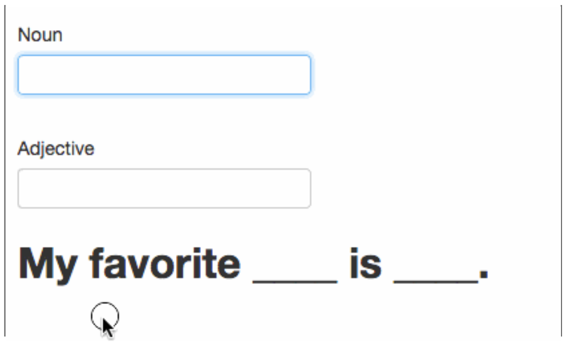
DATA BINDING

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 directive to your app.
4. Add two text input fields to your page.
5. Add an 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



My favorite ____ is ____.

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

is maybe one of the most awesome things about Angular.

```
="beatles=['John', 'Paul', 'George', 'Ringo']">
 The Beatles</h3>
epeat="beatle in beatles">{{beatle}}</li>
```

DEMO

MORE ANGULAR DIRECTIVES

Directive	Description
	The ngRepeat directive instantiates a template once per item from a collection.
	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 .

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

BONUS!

- Use something other than list items for your template. You can use to iterate anything.
- Use 'ngRepeat' to append images to your page, given an array of urls that link to images. (hint: "ng-src")

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.

```
theWho=["Roger Daltry", "Pete Townsend", "Keith Moon"]">
of The Who in uppercase</h3>

eat="member in theWho ">{{ member | uppercase }}</li>
```

DEMO

MORE FILTERS

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

ORDERBY

DEMO

```
theWho=['Roger Daltry', 'Pete Townsend', 'Keith Moon']">
of The Who in alphabetical order</h3>

eat="member in theWho | orderBy:member">{{ member }}</li>
```

```
 The Who in reverse alphabetical order</h3>

t="member in theWho | orderBy:member:true">{{ member }}</li>
```

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 is an array of objects.
- We can access the individual object properties in the array using dot notation.
- We're using 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 .

Again, this example is too big for the slide.

DEMO

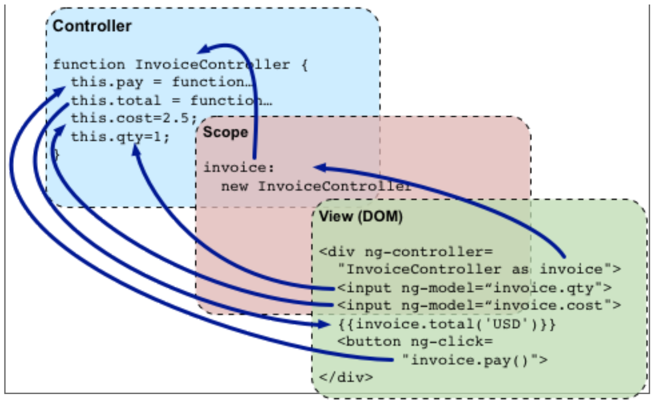
THINGS TO NOTICE:

- Most important: I am actually importing an old version of Angular to make this work.
    - In Angular 1.3+, defining a simple function is no longer sufficient to define a controller.
    - I'm using the older version to demonstrate that a controller is just a function.
- Everything else works exacty as before except our JavaScript is now abstracted into our own 'file'.
- Notice in the controller function we are passing in , now we can finally talk about this.

SCOPE

SCOPE

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



```
function InvoiceController {
  this.pay = function…
  this.total = function…
  this.cost=2.5;
  this.qty=1;
}
```

```
invoice:
  new InvoiceController
```

```
<div ng-controller=
  "InvoiceController as invoice">
  <input ng-model="invoice.qty">
  <input ng-model="invoice.cost">
  {{invoice.total('USD')}}
  <button ng-click=
        "invoice.pay()">
</div>
```

SCOPE

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

```
eController($scope) {
ad = [
rry Garcia', instrument: 'guitar, vocals'},
b Weir', instrument: 'guitar, vocals'},
n \'Pigpen\' McKernan', instrument: 'keyboards, harmonica, vocal
il Lesh', instrument: 'bass, vocals'},
ll 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 component of a software project to be loosely coupled. This makes them easier to test and chang 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.

```
lar.module('myModule', []);
laring a module. More on this in a moment

('myController', function($scope){
r 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

```
yModule" ng-controller="myController">

epeat="beatle in beatles">{{beatle}}</li>
```

```
lar.module('myModule', []);
laring a module. More on this in a moment

('myController', function($scope){
es = ['John', 'Paul', 'George', 'Ringo'];
```

LAB 9

FUNTIMES WITH CONTROLLERS

INSTRUCTIONS

Remake the shopping list program from before, this time using Angular directives, filters and controllers.

- Create an HTML page with two input fields (Item and Price) and an "add" button.
- When the user puts an item name and price in the fields and clicks add:
  - The info from the input fields is added to the list.
  - The total updates.
- Format money values as currency.
- List should be filterable by name and price.

NOTES

- Use separate files for HTML and JavaScript.
- You'll need to make a module. Use the line from the previous examples.

NEW CONTENT AHEAD!
GRANT CHIRPUS

MODULES & SERVICES

GOALS FOR THIS UNIT

1. Review
2. Modules
3. Routes
   - Views
4. Services

MODULES

MODULES

A container for the different parts of an app including controllers, services, filters, directives. Modules are the building blocks that apps are made of. A module's primary job is to serve as place for the pieces of an app to be registered.

MODULE EXAMPLE

We've seen this before. Let's look in more detail

```
lar.module('myCoolModule', []);
th no dependencies.

lar.module('myOtherCoolModule', ['ngRoute']);
th one dependencies
```

MODULE EXAMPLE

The difference between creation vs. retrieval

```
lar.module('myCoolModule', []);
 creates a module. It's sort of like a setter*.

lar.module('myCoolModule');
 retrieves an already created module. It's a getter.

lar.module('myOtherCoolModule');
hrow an error because this module hasn't been created.
```

MODULES

Once you have defined a module, you can add other pieces to it such as controllers, services, custom filters, or directives.

MODULES

```
lar.module('myCoolModule', []);

('myCoolController', function(){
roller stuff


yCoolFactory', function(){
ory stuff
```

ROUTES

ROUTING

In a multipage site we can link to different pages using the same anchor tags and attributes. For a single page application, we need a way to tell the browser to load different content. We can do this with routing.

ROUTING

Originally routing was built-in to Angular. Later, it was decided that it should be maintained as its own module and repository. So we must link to it and include it as a module dependency in order to use it.

IMPORTING NGROUTE

Like other stuff, you can download it or use a CDN

```
ib/angular-route.js"></script>
```

```
lar.module('myModule', ['ngRoute']);
```

NGVIEW

is a directive that goes with the service. It will include a rendere template into the main layout (index.html). Every time the current route changes, the included view changes with it according to the configuration of the service.

```
"></div>
```

```
view>
```

CONFIGURING ROUTES

For our routes, we will add a object to our module. Inside this config object we will inject the to define our routes and define which controllers are used with what views.

The function takes a route name as a string and an object with the route's properties such as and .

The function defines what the router should do for unknown routes.

DEMO

CONFIGURING ROUTES

```
lar.module('gratefulDead', ['ngRoute']);

ction($routeProvider) {
er


ller: 'SimpleController',
teUrl: 'partials/view1.html'

ew2',

ller: 'SimpleController',
teUrl: 'partials/view2.html'

({ redirectTo: '/' });
```

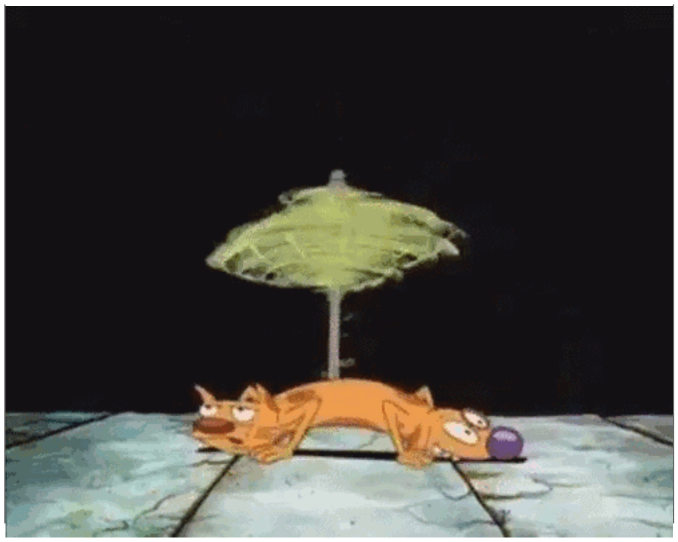
VIEWS

VIEWS

A view is the visible part of the webste (the DOM). When using a router like , views are usually partial (incomplete) snippets of HTML that are injecte into the viewport as needed.

LAB 10

MODULES AND ROUTING



CAT-DOG

INSTRUCTIONS

Create a single page app with two views.

- Create a new project. (index.html, script.js, two more partial views)
- Define a module & controller for this project.
- Remember to download and include .
- Define two separate routes, and an route.
- Set up a shell page with navigation for moving between both views.
- In one route, show a picture of a dog. The other, should show a cat.

NEW CONTENT AHEAD!
GRANT CHIRPUS

SERVICES

SERVICES

Services are reusable components that are separated from views and can be used across multiple controllers and views. There are lots of ways to define services (they are just functions). For now we will define them simply as functions that return objects. We will look at some more conventional patterns later that can also be used

A NOTE ON FACTORIES

There is a subtle difference between services and factories. You may hear the word "factory" thrown around, but we'll get into the difference between the two later in th course because they do almost the same thing. We'll start out by focusing on service

BUILT-IN SERVICES

There are also a number of built-in services. One of the most useful is which provides us access to the browsers XMLHttpRequest object and allows us to make AJAX calls to remote sources.

SERVICES

```
lar.module('myModule');

yService', function(){
ogic
```

```
js
lar.module('myModule', []);

('myController', function(myService){
r logic using myService
```

DEMO

LAB 11

MODULES, ROUTES, & CONTROLLERS

MODULES, ROUTES, & CONTROLLERS

Make a web application that allows a user to enter information in one view and see all of that information displayed and formatted in another.

INSTRUCTIONS

- Create two views: A form with inputs, and a display page.
- Set up routes for both views, provide some navigation solution to move between each.
- In the form view, ask the user for at least two pieces of data about a single item (i. name / instrument in our band member example).
- Each view should have their own controller
- When the user submits the form data, persist it so that it may be used for multipl controllers.
- In the display view, format and display the data.

DEMOS!

BONUS EXERCISE

WORKING WITH EXTERNAL DATA

AJAX

One of the most common operations for web applications is to make API calls to external sources. This is commonly called AJAX requests, though that name is not entirely accurate anymore (we're using JSON mostly now).

INSTRUCTIONS

- Expand the last lab to include external data.
- Use the $http service to pull in external data.
- Use the data from **http://www.reddit.com/r/aww/.json**.
- Display the data somewhere in your information view.
- You may need to use ng-repeat to display multiple things.

- You'll need to get used to parsing through the body of the json to get to the data you want.

- EXTRA BONUS - Allow a user to cycle through the images one at a time on a button click. If they see one they want to "save", allow them to do so with another button, and persist the saved image on the page.

- EXTRA MEGA BONUS - Use a service to serparte the above functionality into separate views.

NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

DIRECTIVES

DIRECTIVES

Directives are the most important and most powerful part of Angular. Directives are the part of angular that allows us to extend native HTML with custom elements and attributes. The result is that this makes our markup a much more expressive and easier to follow.

COMPARE

```
ntainer">
der">
="nav">
ss="nav-item active"><a href="/home">Home</a></div>
ss="nav-item"><a href="/about">About</a></div>
ss="nav-item"><a href="/register">Register</a></div>
ss="nav-item"><a href="/settings">Settings</a></div>
```

VS.

```
>


m>Home</nav-item>
m>About</nav-item>
m>Register</nav-item>
m>Settings</nav-item>


r>
```

DIRECTIVES

We've already made extensive use of directives in Angular. But we've only been usin the ones that come pre-packaged with the framework. It is also possible to create ou own directives to achieve more specific and custom functionality. At their heart, directives are the ability in angular to teach HTML new tricks.

HOW TO DIRECTIVE

To create a custom directive, we start by declaring it in much the same way we do a service or controller and providing a callback

```
lar.module('myModule', []);
'helloWorld', function(){
```

HOW TO DIRECTIVE

Inside the callback function, return an object literal (directive definition) that has a set of properties that will configure the directive.

```
'helloWorld', function(){

"E",
"<h1>Hello World</h1>",
alse
```

DIRECTIVE PROPERTIES

Property	Description
	Defines how the directive can be used (A for Attribute, E for element, etc.)
	Defines the HTML that will be used when this directive is compiled and inserted into the DOM
	Determines whether the directive will be replaced with the template.

HOW TO DIRECTIVE

Once you have defined your directive. You can use it in your HTML just like a native element or attribute. So our previous directive example is used thusly.

```
/hello-world>


/hello:world>
```

DEMO

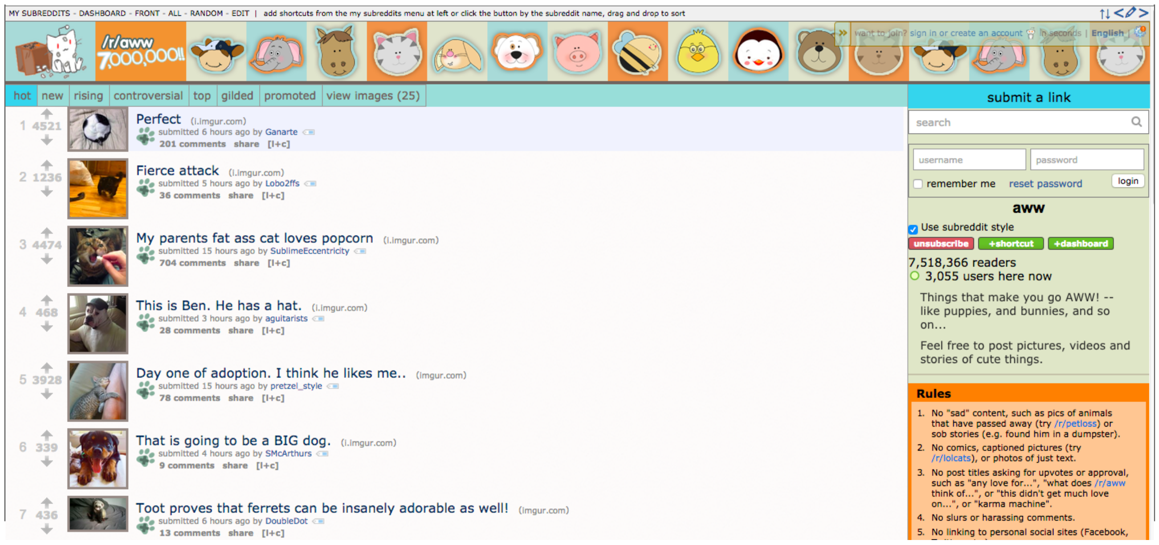
LAB 12

DIRECTIVES

DIRECTIVES

Use basic directives to build a simple expressive html site

- Look at the front page of **reddit/r/aww***
- Remake a similar mock layout (not the content) using directives
- Don't worry about making things clickable. Just make the big sections of the site with directives


MY SUBREDDITS - DASHBOARD - FRONT - ALL - RANDOM - EDIT | add shortcuts from the my subreddits menu at left or click the button by the subreddit name, drag and drop to sort
/r/aww 7,000,000!!
want to join? sign in or create an account in seconds | English |
hot new rising controversial top gilded promoted view images (25)
submit a link
search
username
password
remember me
reset password
login
Use subreddit style
unsubscribe
+shortcut
+dashboard

1 4521 Perfect (i.imgur.com)
submitted 6 hours ago by Ganarte
201 comments share [l+c]

2 1236 Fierce attack (i.imgur.com)
submitted 5 hours ago by Lobo2ffs
36 comments share [l+c]

3 4474 My parents fat ass cat loves popcorn (i.imgur.com)
submitted 15 hours ago by SublimeEccentricity
704 comments share [l+c]

4 468 This is Ben. He has a hat. (i.imgur.com)
submitted 3 hours ago by aguitarists
28 comments share [l+c]

5 3928 Day one of adoption. I think he likes me.. (imgur.com)
submitted 15 hours ago by pretzel_style
78 comments share [l+c]

6 339 That is going to be a BIG dog. (i.imgur.com)
submitted 4 hours ago by SMcArthurs
9 comments share [l+c]

7 436 Toot proves that ferrets can be insanely adorable as well! (imgur.com)
submitted 6 hours ago by DoubleDot
13 comments share [l+c]

aww

7,518,366 readers
3,055 users here now

Things that make you go AWW! -- like puppies, and bunnies, and so on...

Feel free to post pictures, videos and stories of cute things.

Rules

1. No "sad" content, such as pics of animals that have passed away (try /r/petloss) or sob stories (e.g. found him in a dumpster).
2. No comics, captioned pictures (try /r/lolcats), or photos of just text.
3. No post titles asking for upvotes or approval, such as "any love for...", "what does /r/aww think of...", or "this didn't get much love on...", or "karma machine".
4. No slurs or harassing comments.
5. No linking to personal social sites (Facebook,

NEW CONTENT AHEAD!
GRANT CHIRPUS

MORE ON DIRECTIVES

DIRECTIVES, PART 2

If this were all they did, Directives would still be pretty rad. But they actually are capable of a lot more. One of the most powerful aspects of directives is their access t angular's scope object.

LINK FUNCTION

Another of the properties that can be defined for a directive is the property. Th
property is mainly used for attaching event listeners and doing DOM manipulation. To achieve this, the value of the link property is an anonymous function that runs when the directive is compiled.

SIMPLE LINK FUNCTION

By default, directives **share** the scope object with their parent controller.

```
'linkEx', function(){

"E",
"<h1>Hello, {{name}}</h1>",
tion(scope, elem, attrs){
me = 'James!'
```

DEMO

MORE COMPLEX LINK EXAMPLE

The function's main job is to create event handlers and DOM Manipulation (like in jQuery)

In the next example, we bind a property to the scope and use it to change the color of the heading.

In addition, we bind a callback function to the event which changes th mouse cursor.

```
'colorText', function() {

'E',
rue,
'<h1 style="color:{{color}}">Hello World</h1>',
tion(scope, elem, attrs) {
d('click', function() {
ss('color', 'black');
$apply(function() {
e.color = "black";


d('mouseover', function() {
ss('cursor', 'crosshair');
```

DEMO

ANATOMY OF THE LINK FUNCTION

The link function accepts three arguments.

Argument	Description
	The scope of the directive as defined by the directive definition object
	The jQLite (a subset of jQuery) wrapped element on which the directive is applied
	An object of normalized attributes on which the directive is applied

DEEPER INTO DIRECTIVES

OTHER PROPERTIES

This is not an exhaustive list, but here are some more properties that are useful in tl creation of custom directives.

Prop	Description
	Provides a path to an html file instead of writing the template in place
	Allows a directive to include content from another template
	Gives the ability to modify the scope of the directive

TEMPLATE URL

allows us to specify a path to a file for our template instead of hard coding a template right in the directive itself. This is typically a better practice, especially as the complexity of templates increases.

```
'awesomeDir', function() {

'E',
rue,
'partials/awesome-dir.html'
```

TRANSCLUDE

Transclude is a fancy-sounding word that just allows a directive to allow content from another scope to appear with in it. The best metaphor I've ever heard for transclusion directives is to think of them as a picture frame with the directive making up the fram of the picture. The 'foreign' content is what shows up in the center of the picture and it can be included from an entirely different scope.

TRANSCLUDE EXAMPLE

```
'yesTransclude', function() {

: true,
'<div>An example of more things <ng-transclude> </ng-transclude>
rue
```

DEMO

DIRECTIVE SCOPE

By default directives share its parent's scope. We don't always don't want that. If ou directive needs to add properties or functions for its own use, we don't want those properties and functions polluting the parent scope. We have a couple of options to mitigate this:

- A child scope - A scope that inherits the parent scope.
- An isolated scope - A new scope that does not inherit from the parent and exists o its own.

DIRECTIVE SCOPE

In order to create these separate scopes, we use the property in our directive definition object.

```
'childScope', function() {

e,
'<p>This directive will have an inherited scope.</p>'
```

DIRECTIVE SCOPE

To create a scope that is completely isolated from its parent...

```
'childScope', function() {

'<p>This directive will have an isolated scope.</p>'
```

ISOLATE SCOPE WHY?!

Isolating the directive's scope makes it easier to plug in multiple locations throughout your app without having to worry about what scope it will inherit and what will be accessible to it. This does not mean there's no way for an isolated scope to communicate with other components and scopes. But making that work is a prett advanced subject. For now we'll stop here and cover the more advanced stuff on an as-needed basis.

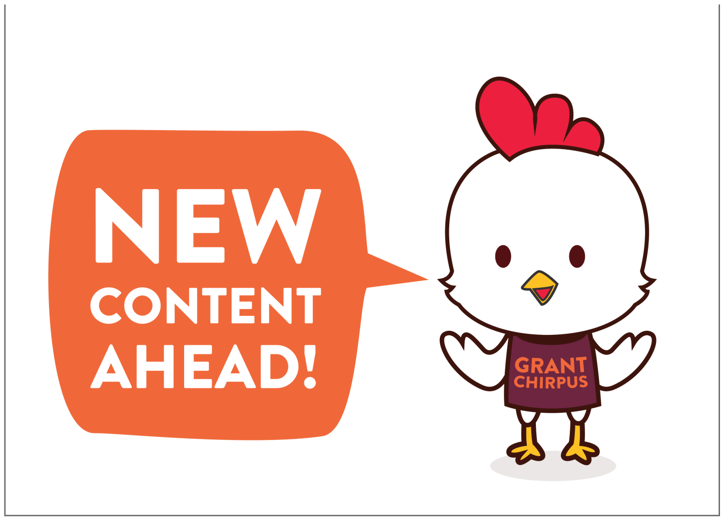
LAB 13

DIRECTIVES PART 2

DIRECTIVES REDUX

Re-make the 'grid of photos' website from week 1

- Build a 'Picture Box' directive
- To make things easy, hard code an object on the scope with the image path / caption for each box
- Iterate the directive on the page using ng-repeat
- BONUS: Make the directive clickable to do something for each directive (ex. display the caption in an alert box)
- BONUS++: Integrate ui-bootstrap and get the caption to show up in a modal


NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

LAB 14

MAKE ME AN ANGULAR APP

INSTRUCTIONS

- Your app must include:
  - Multiple routes (at least 2)
  - Each view should have its own controller
  - At least 2 services:
    - One that works internally for you app
    - One that pulls data from an external source (pick an API, ask Ben or Drew for a recommendation)
  - At least 2 custom directives
  - Inject at least one external dependency (other than ``)
  - Make use of ng-repeat

TESTING

PIZZA TESTS

- Using TDD, build out the back-end code for a pizza ordering program
- The app should allow for the ordering of a Small, Medium or Large pizza
- It should allow a user to choose deep dish, round or thin crust pizza
- It should allow a user to choose three toppings
- It should provide a total cost of the pizza
- The cost of the toppings should change depending on the pizza's size, $1.00 each for small, $2.00 each for medium, $3.00 for large