

Frontend Frameworks

SWE 432, Fall 2016

Design and Implementation of Software for the Web



Today

- How do we build a single page app without dying?
 - MVC/MVVM
 - (AngularJS)

For further reading:

Book: Learning Javascript Design Patterns, Osman (Safari books online)

Book: AngularJS in Action, Ruebelke & Ford (Safari books online)

Book: Learning AngularJS, Williamson (Safari books online)

Only look at references for AngularJS (NOT Angular2)

Demos (source/clone git): <https://github.com/gmu-swe432/lecture10demos>

Source (run in browser): <https://gmu-swe432.github.io/lecture10demos/>

Single Page Application

Many challenges...

Browser



History Tracking

Web Server

Handling a ton of async

Database

Data Binding

Loading Views

Routing

HTTP Request

HTTP Response (JSON)

Persistence tier



AngularJS — Superheroic JavaScript Framework

<https://angularjs.org>

ANGULARJS Home Learn Develop Discuss Search

AngularJS

by Google

HTML enhanced for web apps!

[Download AngularJS 1](#)
(1.5.8 / 1.2.30)

Try the new Angular 2

[View on GitHub](#) [Design Docs & Notes](#)

Follow +AngularJS on  [Follow @angularjs](#) 173K followers [Tweet](#)

 Learn Angular in your browser for free!

Angular to the Rescue

- Full-featured SPA framework (can be used for non-SPA sites too!)
- It's full of buzzwords!
 - Data binding, MVC, MVVM, Routing, Testing, jqLite, Templates, History, Factories, Directives, Services, Dependency Injection, Validation, and all of their friends!
- There are other frameworks too, they work fine, but we're focusing on AngularJS:
 - Aurelia
 - Backbone.js
 - Ember.js

Keeping stuff
organized... How do we
break apart components?

My Very Cool Drink Factory (MVC)



The MVC Drink Factory: A recipe

- Requires:
 - 3oz coconut milk
 - 3oz almond milk
 - 2 frozen bananas
 - 2 tbsp peanut butter
 - 1 tbsp agave nectar
- Place all ingredients in blender and blend for 45 seconds
- Serve in a pint glass, garnish with banana slice, cherry, and whipped cream (optional)



The MVC Drink Factory: Abstract

- What makes a drink?
 - Ingredients
 - Glasswear
 - Recipe
- Recipe *controls* the entire process
- Ingredients make up the content of the drink
- Glass changes how you see the drink, but not its contents

The MVC Drink Factory

- Can make *other* drinks by changing the ingredients, keeping the steps to follow and the glass
- Can make also keep the ingredients and steps, change the presentation



Same recipe, different presentation



Same recipe, different ingredients

The MVC Modular Drink Factory

- My Very Cool factory separates concerns between recipes, ingredients, and glasses
- Different people can pull out ingredients, follow the recipe, and pour the result into the correct glass without knowing exactly what the other person does
- Could even completely replace how the ingredients are gathered (maybe use pre-portioned), and it doesn't effect the rest of the process



My Very Cool Drink Factory

- Wow, this separation of concerns is just what we want in our web apps!
- Because it's so modular, we named an entire design pattern based on this recipe, ingredient, presentation pattern (MVC)
- Alternatively, we might call it **Model-View-Controller**
 - Model: Ingredients
 - Controller: Recipe
 - View: Glass/presentation

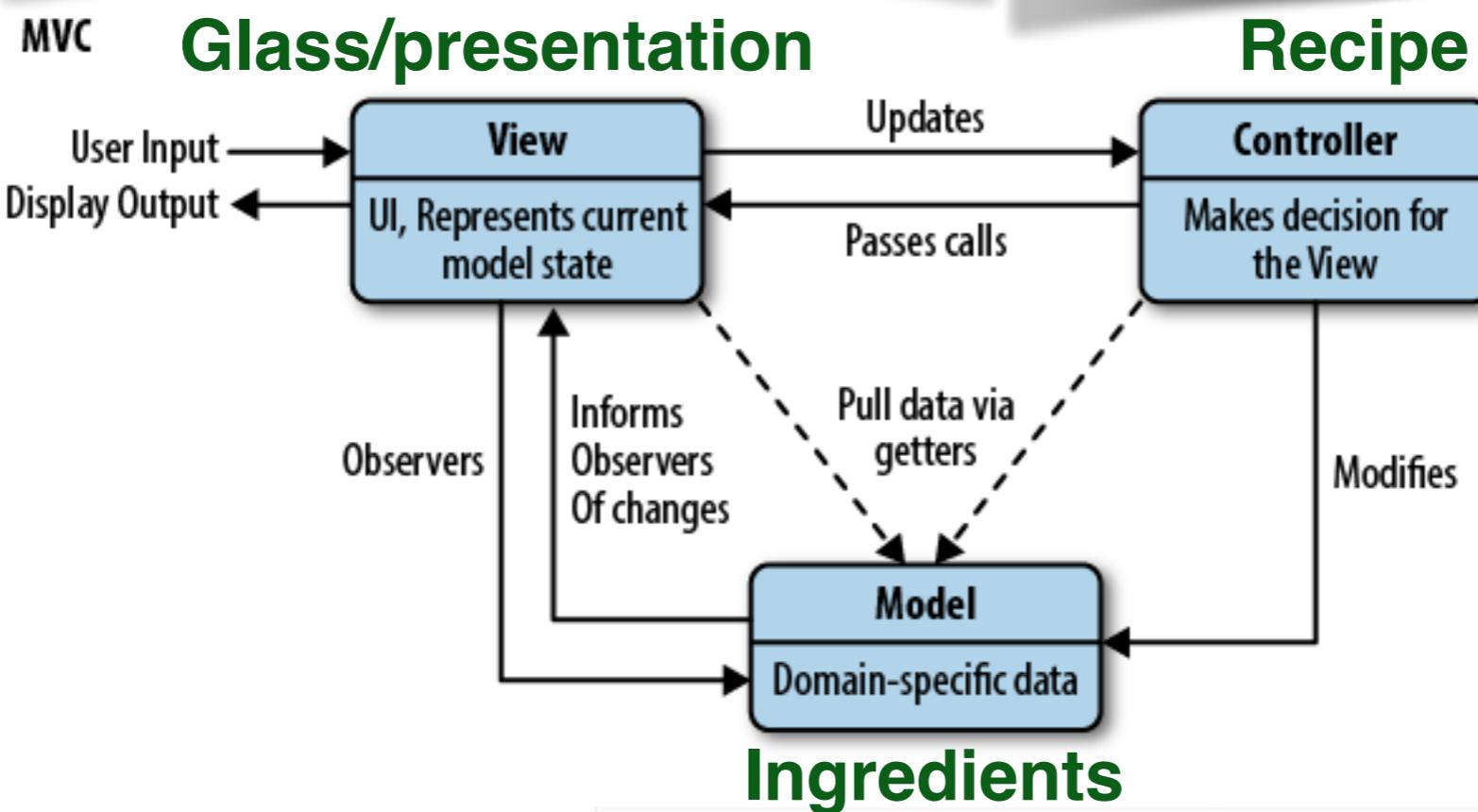
MV* Patterns

- The mother of them all: MVC
 - Originally from 70's: UIs were just becoming possible... how to separate presentation from data and logic?
 - Model: domain-specific data, doesn't matter how it's interacted with
 - View: visual representation of current state of model
 - Controller: Moderates user interactions, makes business decisions
 - *Separation of concerns*

MVC & JavaScript

DOM templates

JS that receives input from DOM, deploys spinner, etc.

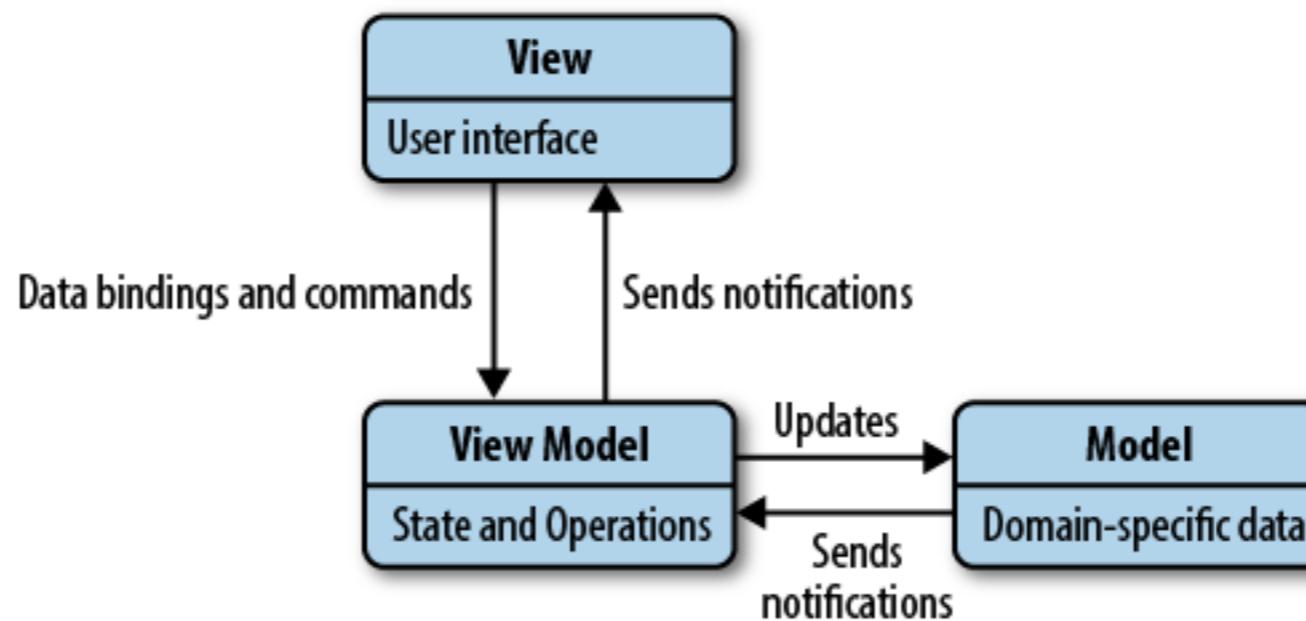


Firebase todoRef list

Firebase callbacks that update view directly

***Note that in drink factory, the glass doesn't care about the ingredients**

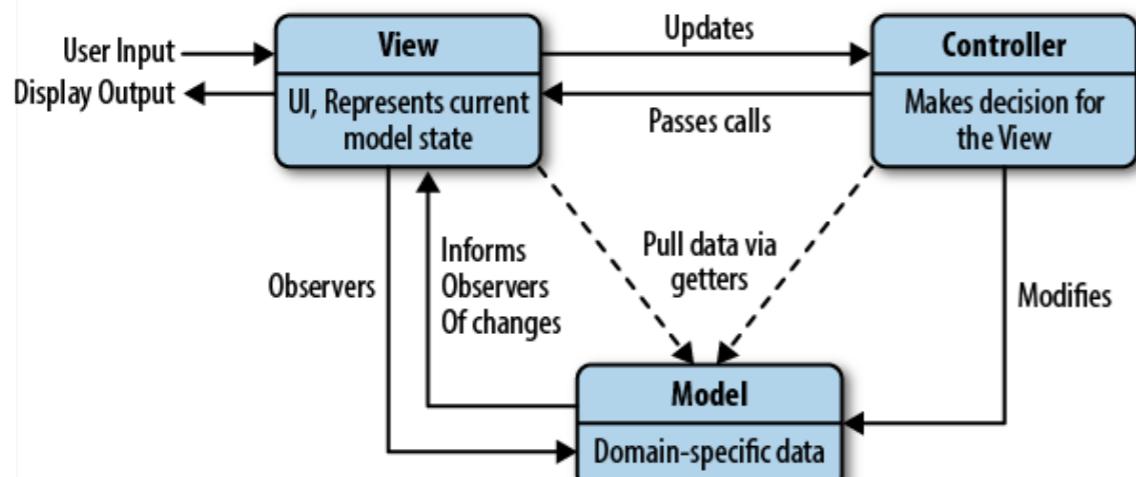
MVVM



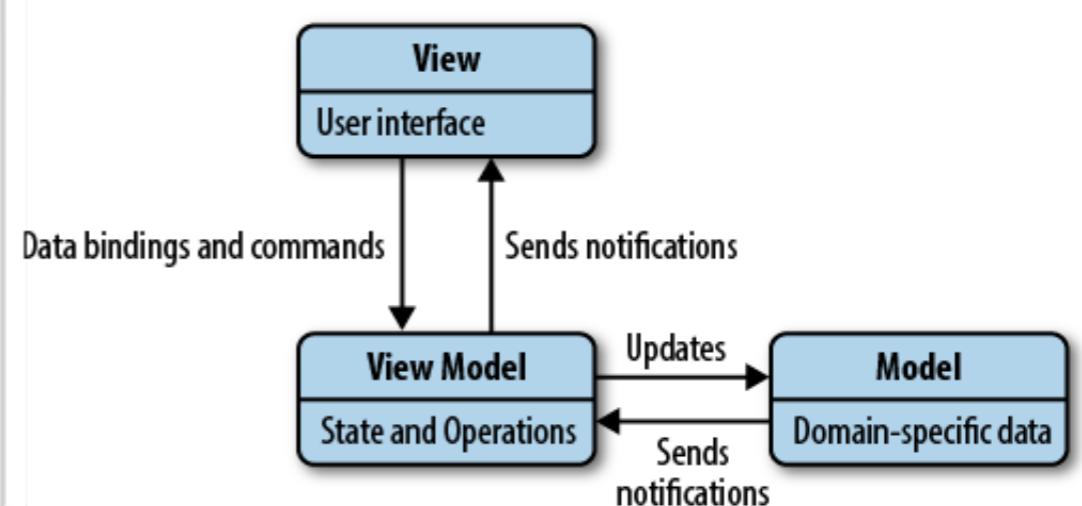
- View does not communicate with model directly
- Models are much more dumb: no formatting, etc
- ViewModel: like a controller from MVC, but only does data translation/formatting between M-V
- More directly maps to MyVeryCool Drink Factory than MVC does

MVC vs MVVM

MVC



MVVM



Low level controller/model code can be easily shared (especially in server apps)

Views can have direct access to model

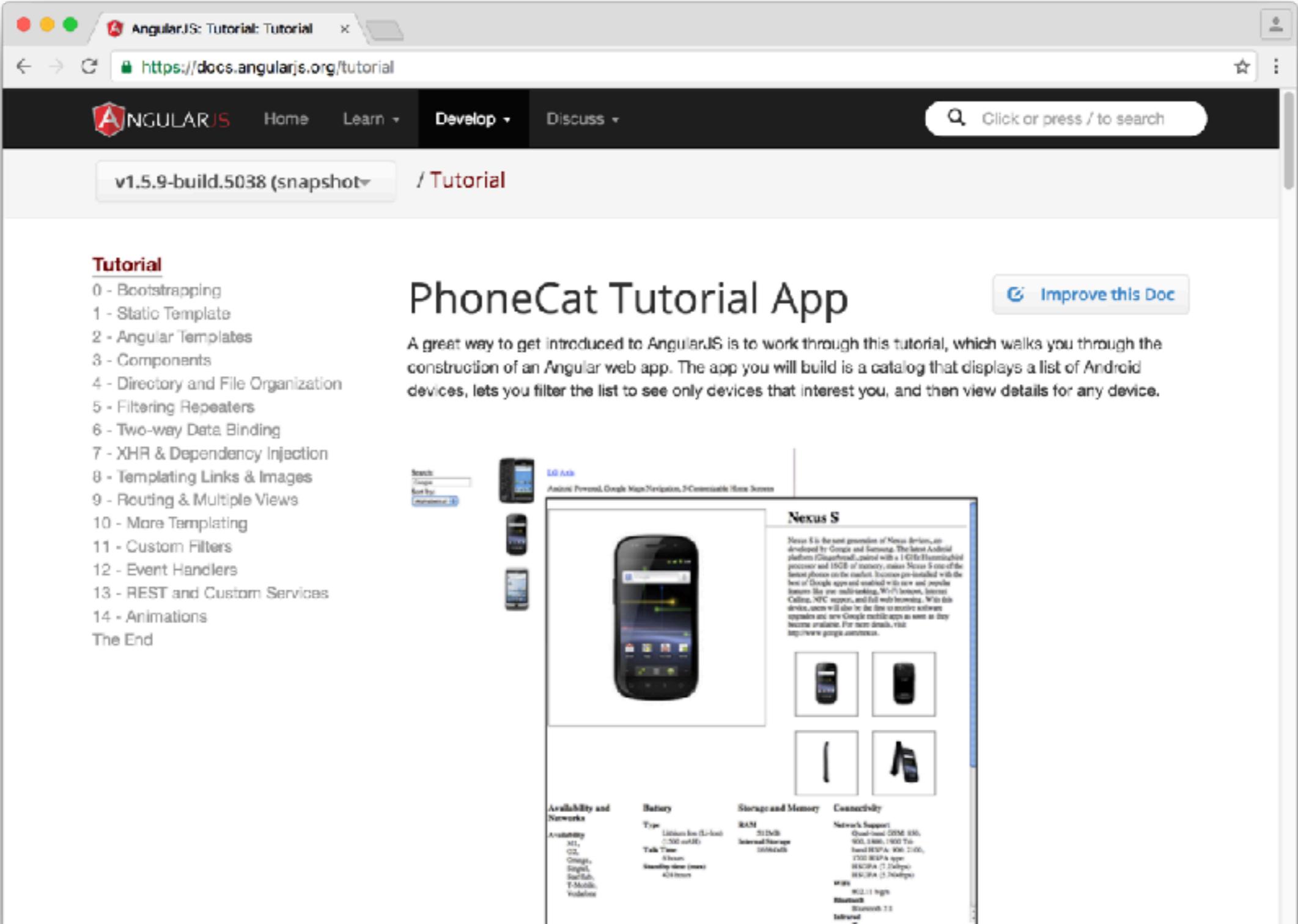
Easier to develop in parallel (V only talks to VM)

View is completely “dumb” and just needs data bindings

AngularJS

- Supports MVC/MVVM
- Provides structure to organize your code
- Two-way data binding
- Uses plain old objects for your data - no fancy structures needed
- HTML templating (like react)
- Designed for SPAs

Angular Documentation: Great



The screenshot shows a web browser displaying the AngularJS tutorial page at <https://docs.angularjs.org/tutorial>. The page title is "AngularJS: Tutorial: Tutorial". The navigation bar includes links for "Home", "Learn", "Develop", "Discuss", and a search bar. A dropdown menu shows "v1.5.9-build.5038 (snapshot)". The main content is titled "PhoneCat Tutorial App". It describes the tutorial as a way to get introduced to AngularJS by building a catalog of Android devices. Below this, there is a large image of an LG AX3 smartphone displaying the PhoneCat app interface. To the right of the phone, there is a detailed specification table for the "Nexus S" device, including sections for Availability and Networks, Battery, Storage and Memory, and Connectivity. The table lists various technical specifications for the Nexus S.

Availability and Networks	Battery	Storage and Memory	Connectivity
Availability: AT&T, Cingular, Cricket, Google, Sprint, T-Mobile, Verizon	Type: Lithium Ion (Li-Ion) (3.7V 1800 mAh) Talk Time: 8 hours Standby time (max): 424 hours	RAM: 512 MB Internal Storage: 16GB/32GB	Network Support: Quad-band GSM 850, 900, 1800, 1900 MHz 3G: HSDPA 106, 2100, 1200, 1400 Mbps HSPA+ (7.2 Mbps) HSUPA (5.76 Mbps) WIFI: 802.11 b/g/n Bluetooth: Bluetooth 2.1 Infrared

Directives & Data Binding

- Core feature of Angular
- Unlike React (add HTML to code), Angular lets us *direct* the html to have some code in it too
- Lets us add code into HTML
- Angular example:

```
<!DOCTYPE html>
<html lang="en" ng-app> ←
<head>
  <meta charset="UTF-8">
  <title>My Angular Demo</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/
angular.js"></script>
```

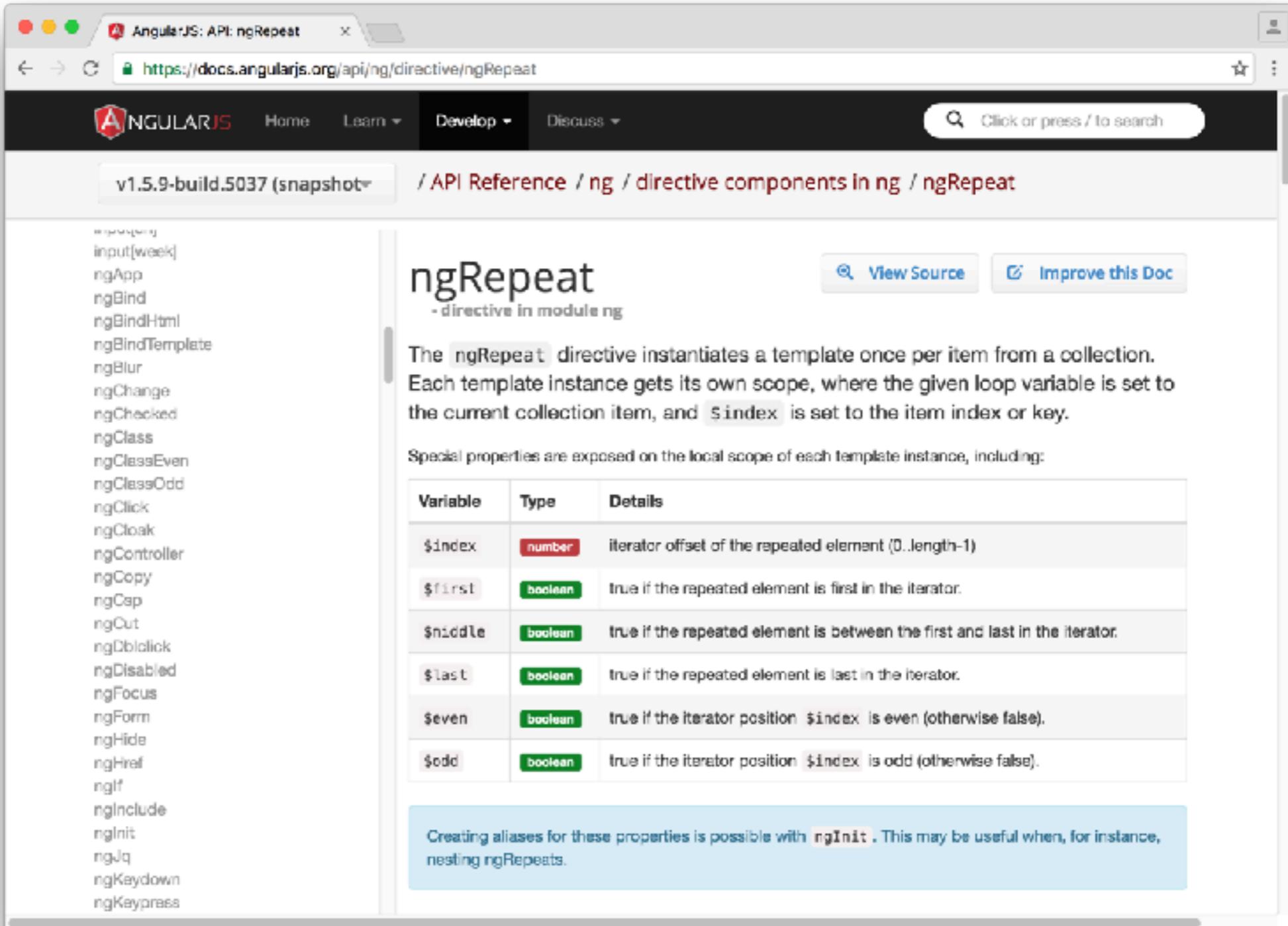
```
</head>
<body>
What's your name? <input type="text" data-ng-model="name" /> {{name}}
</body>
</html>
```

Directives

Data Binding

Simple Data Binding Example

Other Directives



The screenshot shows a web browser displaying the AngularJS API documentation for the `ngRepeat` directive. The URL in the address bar is `https://docs.angularjs.org/api/ng/directive/ngRepeat`. The page title is "AngularJS: API: ngRepeat". The main content area is titled "ngRepeat" and describes it as a directive in the `ng` module. It explains that the `ngRepeat` directive instantiates a template once per item from a collection, creating a new scope for each template instance. The `$index` variable is set to the current collection item, and `$index` is set to the item index or key. A table lists special properties exposed on the local scope of each template instance, including `$index`, `$first`, `$middle`, `$last`, `$even`, and `$odd`. A note at the bottom states that creating aliases for these properties is possible with `ngInit`, which may be useful for nesting `ngRepeats`. The left sidebar contains a list of other AngularJS directives, and the top navigation bar includes links for Home, Learn, Develop, Discuss, and a search bar.

ngRepeat

- directive in module ng

The `ngRepeat` directive instantiates a template once per item from a collection. Each template instance gets its own scope, where the given loop variable is set to the current collection item, and `$index` is set to the item index or key.

Special properties are exposed on the local scope of each template instance, including:

Variable	Type	Details
<code>\$index</code>	number	iterator offset of the repeated element (0..length-1)
<code>\$first</code>	boolean	true if the repeated element is first in the iterator.
<code>\$middle</code>	boolean	true if the repeated element is between the first and last in the iterator.
<code>\$last</code>	boolean	true if the repeated element is last in the iterator.
<code>\$even</code>	boolean	true if the iterator position <code>\$index</code> is even (otherwise false).
<code>\$odd</code>	boolean	true if the iterator position <code>\$index</code> is odd (otherwise false).

Creating aliases for these properties is possible with `ngInit`. This may be useful when, for instance, nesting `ngRepeats`.

input[week]
ngApp
ngBind
ngBindHtml
ngBindTemplate
ngBlur
ngChange
ngChecked
ngClass
ngClassEven
ngClassOdd
ngClick
ngCloak
ngController
ngCopy
ngCap
ngCut
ngDblclick
ngDisabled
ngFocus
ngForm
ngHide
ngHref
ngIf
ngInclude
ngInit
ngJq
ngKeydown
ngKeypress

Other Directives

- ng-init
 - Initialize variables within the scope of a DOM element
- ng-repeat
 - Replicate a DOM element over an array

```
<div class="container"
  data-ng-init="names=[ 'Dave', 'Napur', 'Heedy', 'Shriva' ]">

  <h3>Looping with the ng-repeat Directive</h3>
  <ul>
    <li data-ng-repeat="name in names">{{ name }}</li>
  </ul>
</div>
```

Filters

- Allow you to modify the text going into data bindings
- Only want to make simple modifications here
- Syntax:
`{{todo.text | uppercase}}`
 - (Converts the todo to uppercase)
- Other uses:
 - Select only some values in a list
 - Order a list

```
<div ng-repeat="todo in todos | orderBy: '-priority' ">
```

Partials:

- A "Partial" HTML document
- Can be included into another with **<ngInclude>**
- Example:

```
<div>
  What's your name? <input type="text"
  data-ng-model="name"/> Hello, {{name}}!
</div>
```

partials/hello.html

```
<ng-include src="partials/hello.html"></ng-include>
```

index.html

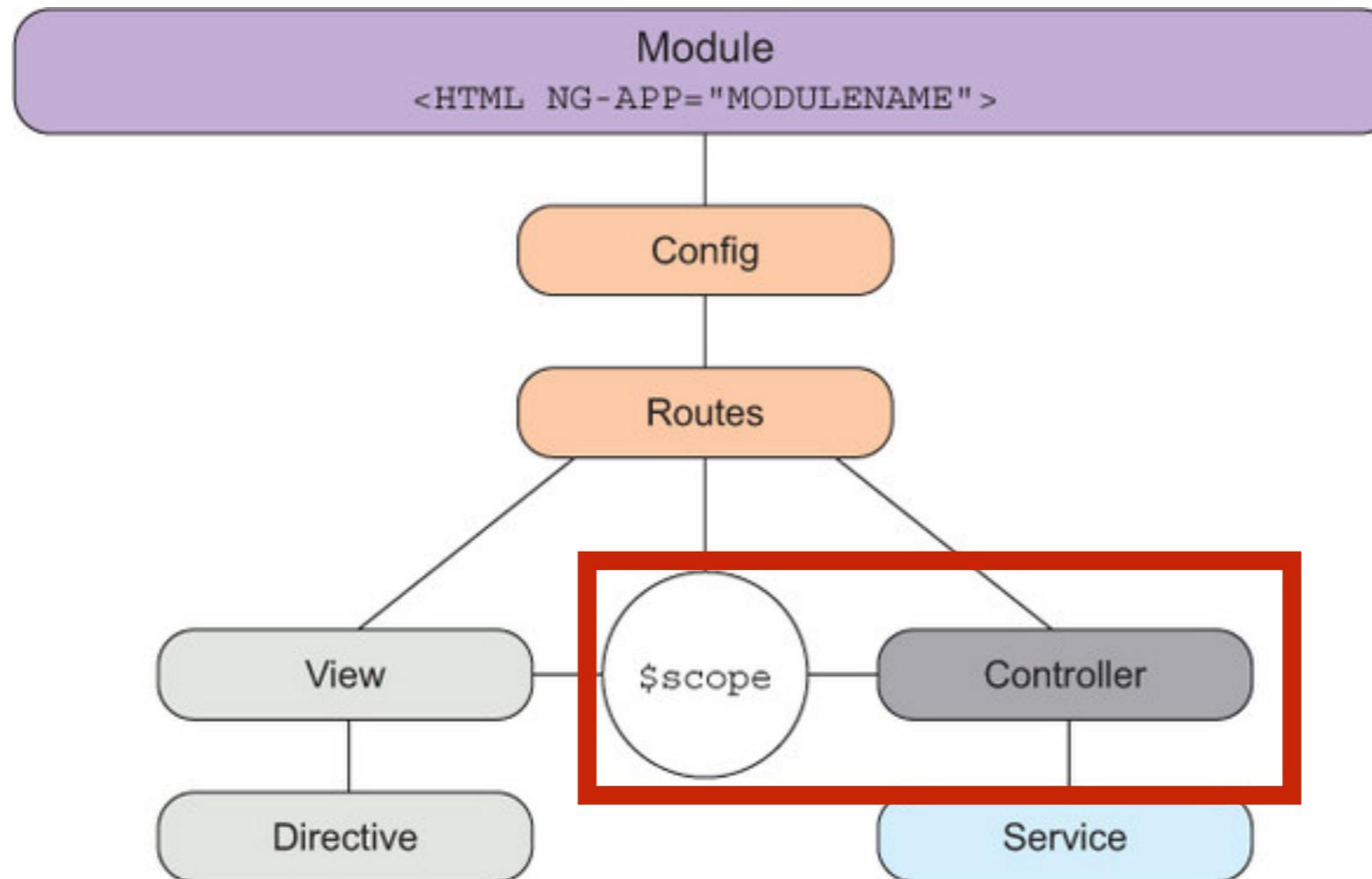
Partials & Components

The screenshot shows a Twitter feed with several components highlighted by arrows:

- User profile partial** (blue arrow): Points to the sidebar on the left, which displays the user profile of Thomas LaToza.
- Who to follow partial** (orange arrow): Points to the "Who to follow" sidebar on the right.
- Follow partial** (orange arrow): Points to the "Follow" button within the "Who to follow" sidebar.
- Feed partial** (green arrow): Points to the main feed area, which contains a promoted tweet from Toyota USA.
- Feed item partial** (purple arrow): Points to the main feed area, which contains a tweet from davideshpherd.

Partial Demo

Views, Controllers, Scopes



- Angular has a lot more than just views and directives
- Let's focus on controllers and scope

Angular Controllers

- Each controller is a function that gets passed `$scope`
- `$scope` is the bridge between the controller and view
- `$scope` is initially empty when the controller is called, and then it sets some properties on it
- When a view uses a controller, it inherits its `$scope`

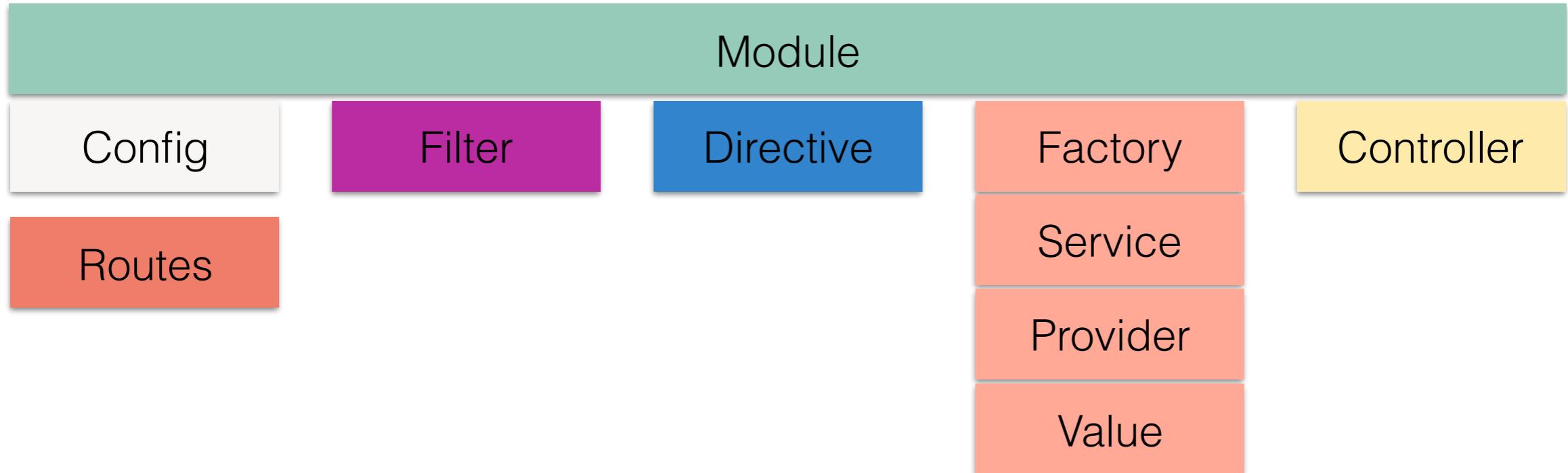
```
function TodoController($scope)
{
  $scope.todos = [
    { text: "Write more demos", priority: 5 },
    { text : "Add some gifs", priority: 10 }
  ];
}
```

Views & Controllers

- Select a controller for a DOM element, and it will provide variables for everything contained in it
- Can have multiple controllers on one page

```
<div class="todoList" data-ng-controller='TodoController'>
  <div ng-repeat='todo in todos | orderBy: "-priority"'>
    {{todo.text | uppercase}}
  </div>
</div>
<script>
  function TodoController($scope)
  {
    $scope.todos = [
      { text: "Write more demos", priority: 5},
      { text : "Add some gifs", priority: 10}
    ];
  }
</script>
```

Modules as Containers



- Modules contain everything that we need for a single component
- Organize views, controllers, etc.
- How do we make and use them?

Creating a Module

- Create a module and add a controller:

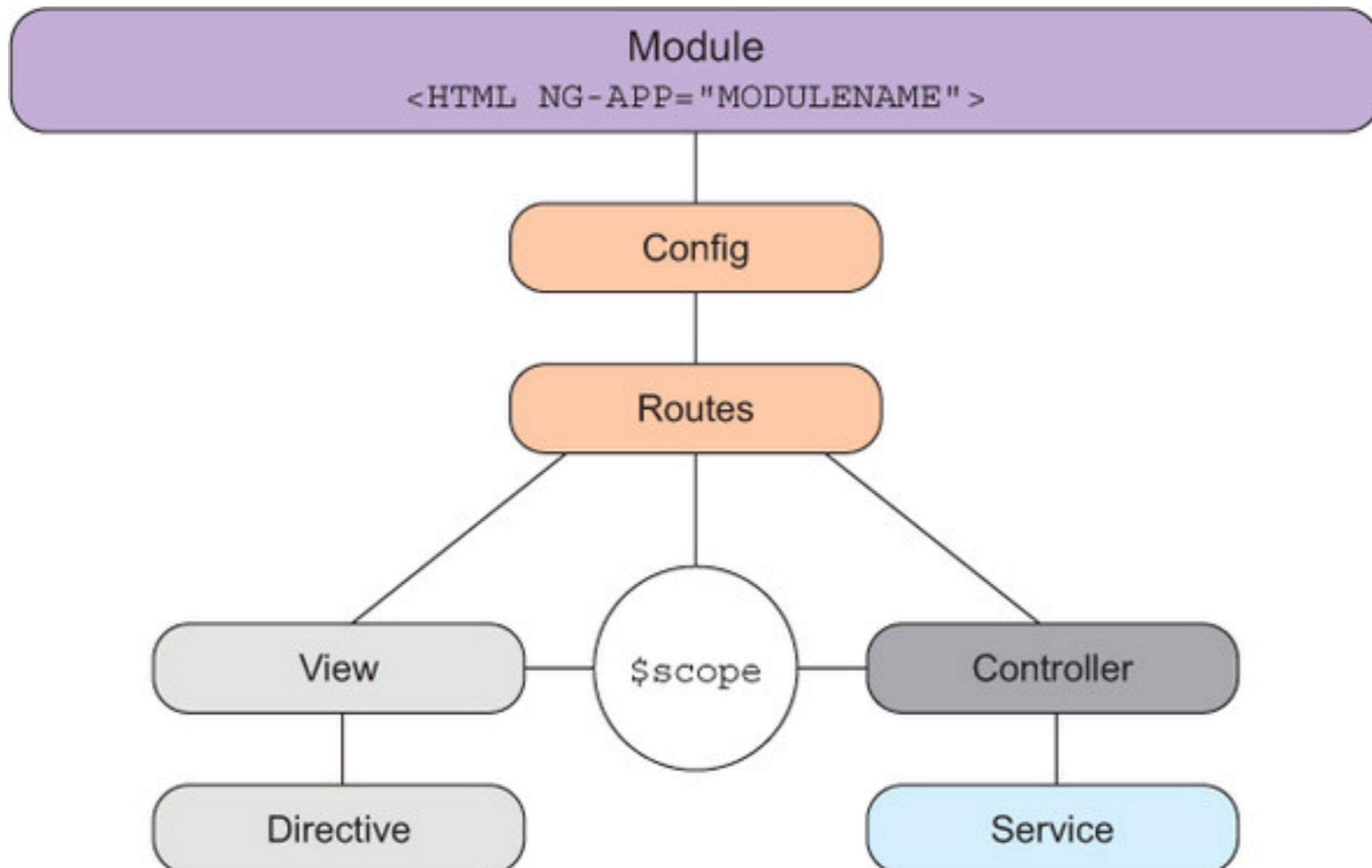
```
var myApp = angular.module('demoApp', []);
myApp.controller("TodoController", TodoController);
```

- The empty array can instead specify dependencies
 - Example dependency (a great one!): **firebase**
 - Controllers should not stand on their own - must be part of module
 - Module name must be the name provided in ng-app

```
<html lang="en" data-ng-app="demoApp">
```

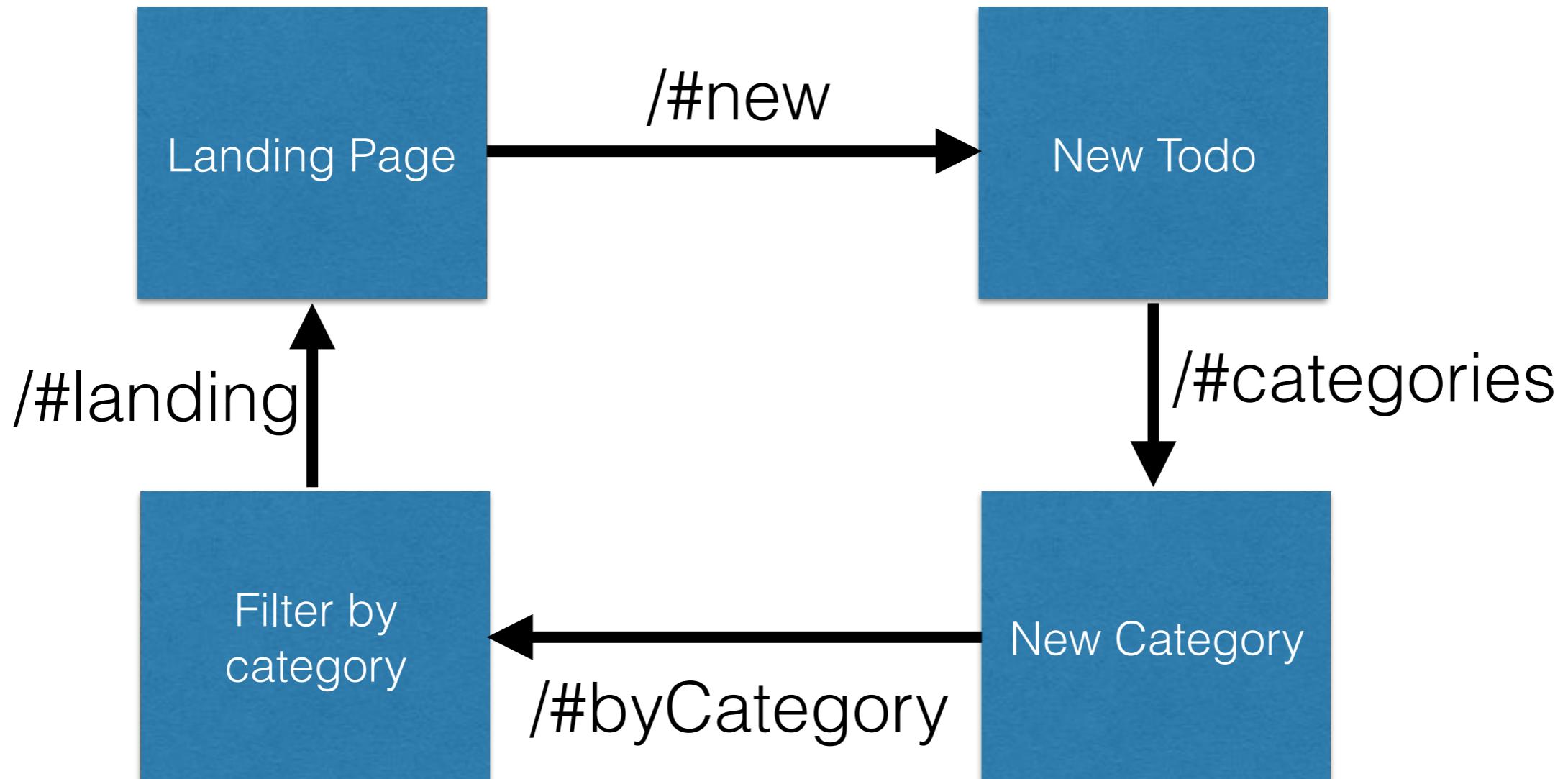
Demo: Modules,
Controllers, Firebase

Modules, Routes, Services



Routes

Routes are paths from view/controllers to others



Routes

- AngularJS makes routes read like magic!

```
myApp.config(function($routeProvider){  
  $routeProvider.when("/", {  
    controller: "TodoController",  
    templateUrl: "partials/editableTodos.html"  
  }).when("/categories", {  
    controller: "CategoryController",  
    templateUrl: "partials/categories.html"  
  }).otherwise({redirectTo: "/"});  
});
```

Dependency injection magic!

- Reads like a sentence (chaining!)

Partials

- Easy way to have "partial" HTML documents and combine them, magically-dynamically into one!
- Will be included by the **route**, into the container labeled with the directive

```
<div data-ng-view></div>
```

```
...
myApp.config(function($routeProvider){
  $routeProvider.when("/",{
    controller: "TodoController",
    templateUrl: "partials/editableTodos.html"
  }).when("/categories",{
    controller: "CategoryController",
    templateUrl: "partials/categories.html"
  }).otherwise({redirectTo: "/"});
});
```

Demo: Routes +
Partials

Exit-Ticket Activity

Go to socrative.com and select “Student Login”

Class: SWE432001 (Prof LaToza) or SWE432002 (Prof Bell)

ID is your @gmu.edu email

- 1: How well did you understand today's material
- 2: What did you learn in today's class?

For question 3: How do you think you will use React in your HW this week?