

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, Osmani (Safari books online)

Book: AngularJS in Action, Ruebbelke & 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...

DOM Manipulation

Browser

```
<doctype html>
<html>
  <title>This is a title</title>
  <body>
    <h1>Hello world!</h1>
  </body>
</html>
```

HTML



HTML elements

```
helloWorld();

function helloWorld() {
  var message = "hi!Hello, world!</h1>";
  $("body").html();
}
```

Javascript

History Tracking

HTTP
Request

HTTP
Response
(JSON)

Caching

Data Binding

Routing

Web Server

Persistence tier

Loading Views

Handling a ton of async

Database





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 **M**odel-**V**iew-**C**ontroller
 - Model: Ingredients
 - Controller: Recipe
 - View: Glass/presentation

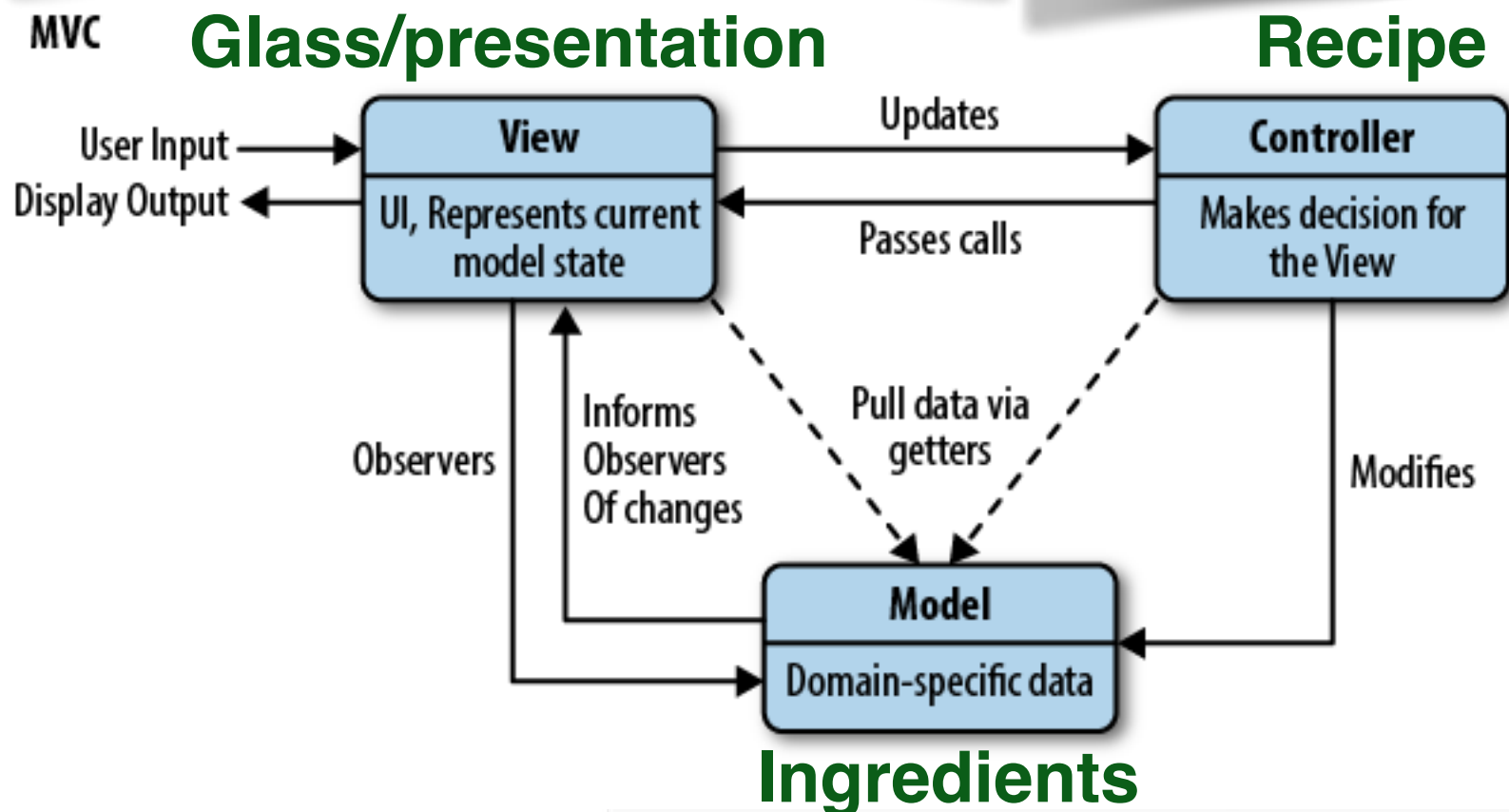
MV* Patterns

- 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.

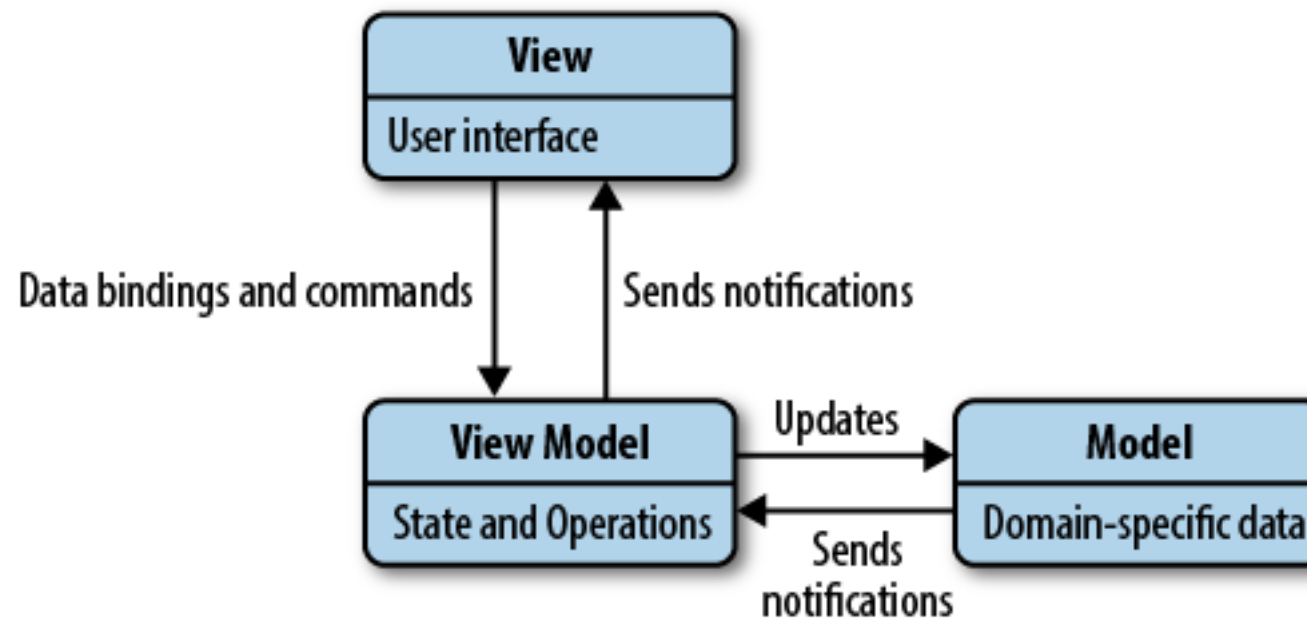


Firestore todoRef list

Firestore callbacks that update view directly

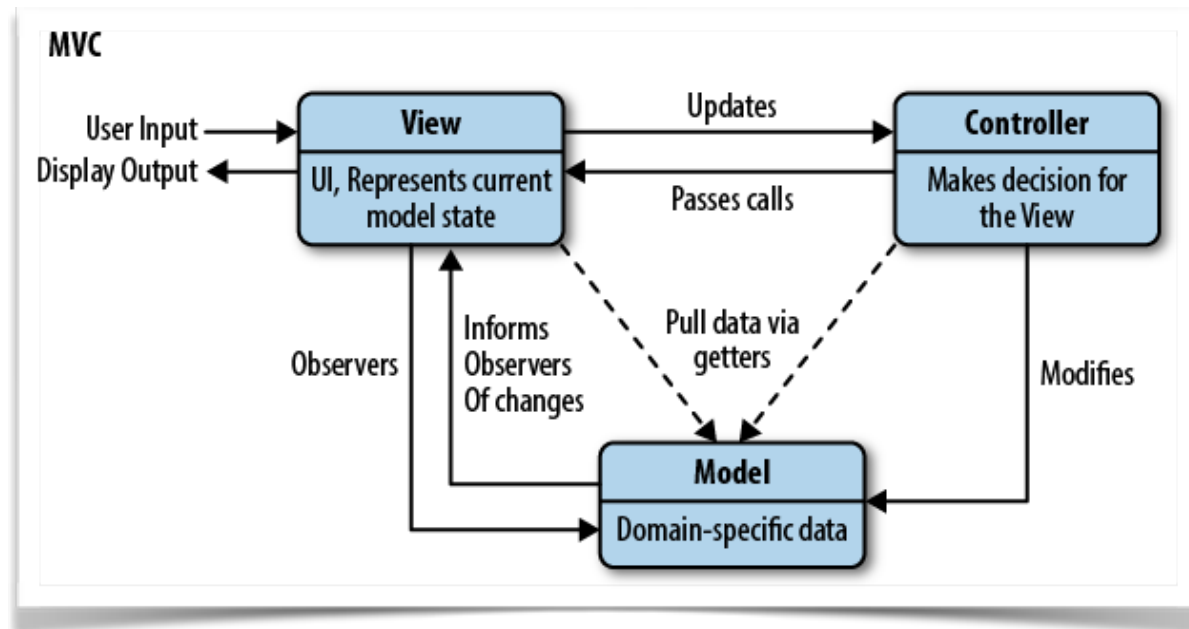
***Note that in drink factory, the glass doesn't care about the ingredients**

MVVM



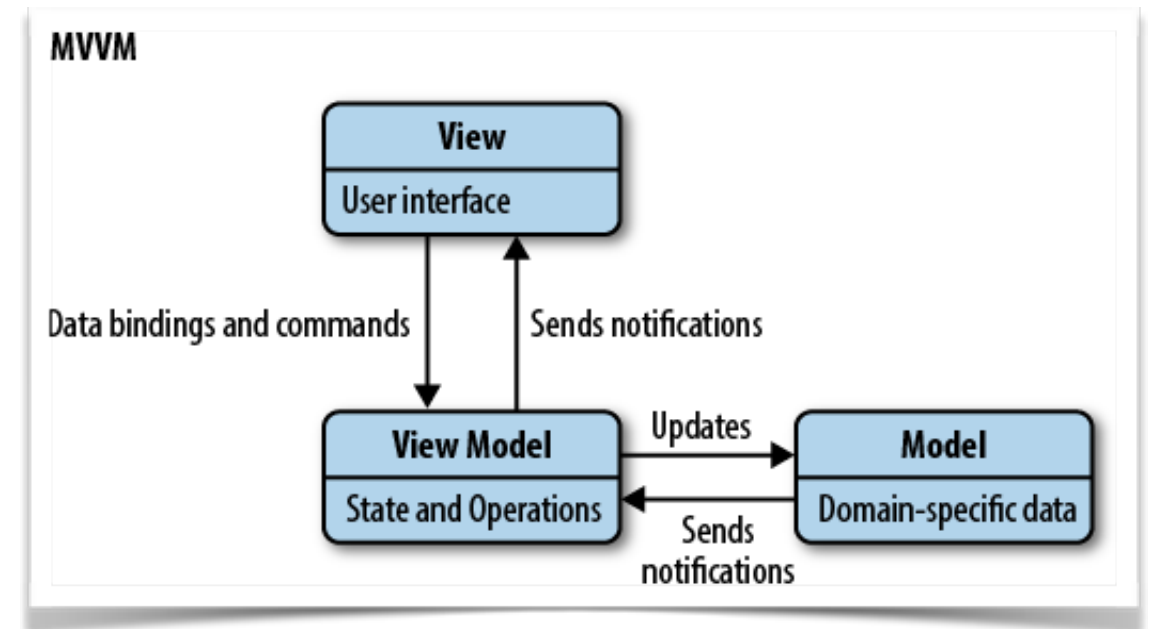
- View does not communicate with model directly
- Models are much more dumb: no formatting, etc
- ViewModel: like a controller from MVC, but only does data translation/formatting between M-V
- More directly maps to MyVeryCool Drink Factory than MVC does

MVC vs MVVM



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 the AngularJS documentation website. The browser address bar displays `https://docs.angularjs.org/tutorial`. The navigation bar includes links for Home, Learn, Develop, and Discuss, along with a search bar. Below the navigation bar, the version `v1.5.9-build.5038 (snapshot)` and the path `/ Tutorial` are shown. The main content area features a sidebar with a table of contents for the tutorial, a main heading for the 'PhoneCat Tutorial App', and a description of the app. The app description states: 'A great way to get introduced to AngularJS is to work through this tutorial, which walks you through the construction of an Angular web app. The app you will build is a catalog that displays a list of Android devices, lets you filter the list to see only devices that interest you, and then view details for any device.' Below the description, there is a search bar and a list of devices. The 'Nexus S' device is highlighted, showing its image and a detailed specification table.

Tutorial

- 0 - Bootstrapping
- 1 - Static Template
- 2 - Angular Templates
- 3 - Components
- 4 - Directory and File Organization
- 5 - Filtering Repeaters
- 6 - Two-way Data Binding
- 7 - XHR & Dependency Injection
- 8 - Templating Links & Images
- 9 - Routing & Multiple Views
- 10 - More Templating
- 11 - Custom Filters
- 12 - Event Handlers
- 13 - REST and Custom Services
- 14 - Animations
- The End

PhoneCat Tutorial App

A great way to get introduced to AngularJS is to work through this tutorial, which walks you through the construction of an Angular web app. The app you will build is a catalog that displays a list of Android devices, lets you filter the list to see only devices that interest you, and then view details for any device.

Search:

Nexus S

Nexus S is the next generation of Nexus devices, developed by Google and Samsung. The latest Android platform (Gingerbread), paired with a 1.4GHz Hummerhead processor and 1GB of memory, makes Nexus S one of the fastest phones on the market. It comes pre-installed with the best of Google apps and enriched with new and popular features like your multi-tasking, Wi-Fi hotspot, Internet Calling, NFC support, and full web browsing. With this device, users will also be the first to receive software upgrades and new Google mobile apps as soon as they become available. For more details, visit <http://www.google.com/nexus>.

Availability and Networks	Battery	Storage and Memory	Connectivity
Availability AT&T, CDMA, Google, Sprint, T-Mobile, Vodafone	Type Lithium Ion (Li-Ion) (1500 mAh) Talk Time 8 hours Standby time (max) 424 hours	RAM 512MB Internal Storage 16GB (with 4GB)	Network Support Quad-band GSM: 850, 900, 1800, 1900 MHz HSPA+: 900, 2100, 2600 MHz Wi-Fi 802.11 b/g/n Bluetooth Bluetooth 2.1+EDR USB Micro-USB

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>
```

The diagram illustrates the relationship between Directives and Data Binding in Angular. A box labeled "Directives" has two arrows: one pointing to the `ng-app` attribute in the `<html>` tag, and another pointing to the `data-ng-model="name"` attribute in the `<input>` tag. A box labeled "Data Binding" has an arrow pointing to the `{{name}}` interpolation in the text "What's your name? {{name}}".

Simple Data Binding Example

Other Directives

The screenshot shows the AngularJS API documentation for the `ngRepeat` directive. The browser address bar shows the URL `https://docs.angularjs.org/api/ng/directive/ngRepeat`. The page header includes the AngularJS logo and navigation links: Home, Learn, Develop, and Discuss. A search bar is also present. 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, and each instance gets its own scope with the loop variable and `$index` set. A table lists special properties exposed on the local scope of each template instance, including `$index`, `$first`, `$middle`, `$last`, `$even`, and `$odd`. A note at the bottom mentions that creating aliases for these properties is possible with `ngInit`.

AngularJS: API: ngRepeat

Home Learn Develop Discuss

Click or press / to search

v1.5.9-build.5037 (snapshot) / API Reference / ng / directive components in ng / ngRepeat

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`.

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)
- `<div ng-repeat="todo in todos | orderBy: '-priority' ">`
 - (Shows all todos ordered by key)
- Other uses:
 - Select only some values in a list
 - Order a list

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

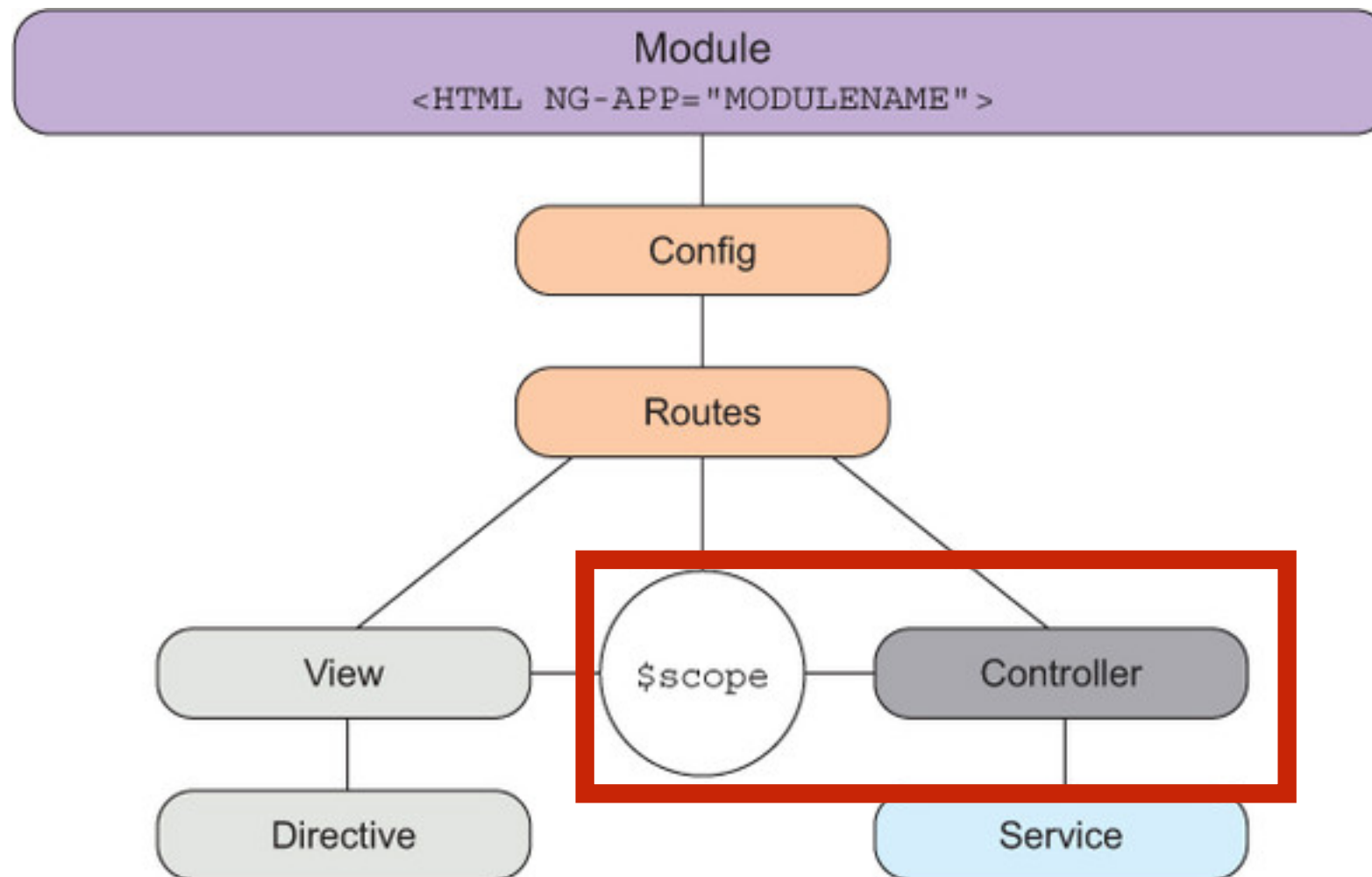
The image shows a Twitter profile page for Thomas LaToza (@ThomasLaToza). The page is annotated with colored boxes and arrows pointing to specific components, labeled as follows:

- User profile partial:** Points to the user's profile header, including the profile picture, name, and bio.
- Who to follow partial:** Points to the 'Who to follow' section, which lists users suggested for following.
- Follow partial:** Points to the 'Follow' button on the 'Who to follow' list.
- Feed partial:** Points to the main content area of the profile, which displays a list of tweets.
- Feed item partial:** Points to a single tweet in the feed, including the user's profile picture, name, bio, and the tweet content.

The Twitter interface includes a top navigation bar with links to Home, Moments, Notifications, and Messages. The left sidebar shows the user's profile statistics (61 tweets, 229 following, 163 followers) and a list of trending topics. The main content area displays a tweet from 'Talks at Google' and a tweet from 'Toyota USA' featuring a blue car. The right sidebar shows a 'Who to follow' section with three users: Grace Lewis, Gregorio Robles, and Loren Tenveen.

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`

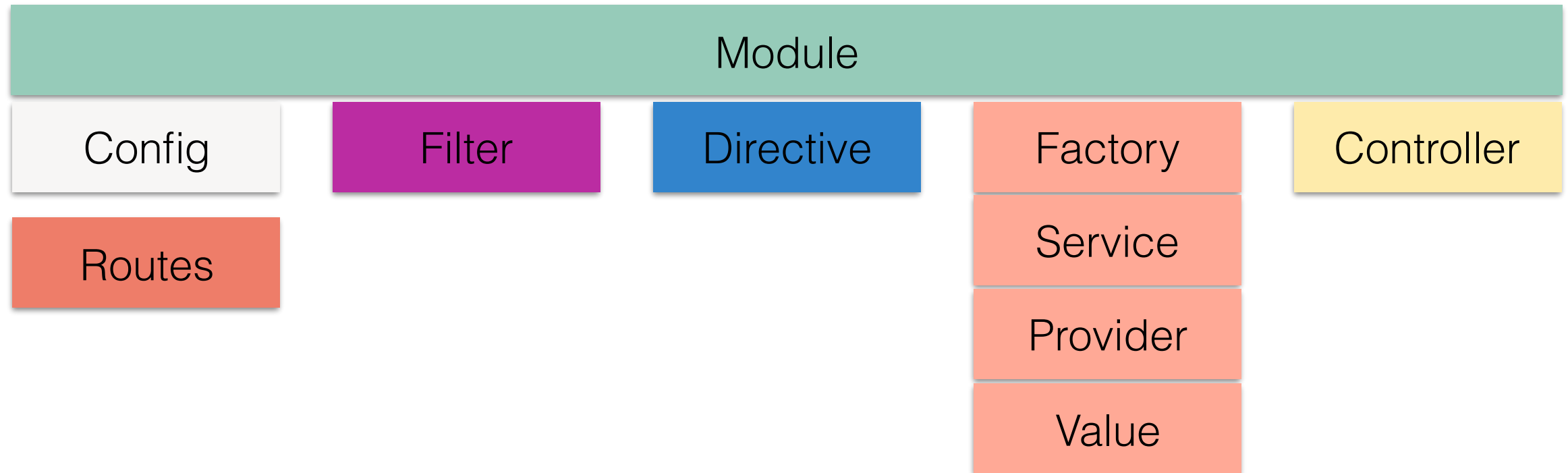
```
function TodoController($scope)
{
    $scope.todos = [
        { text: "Write more demos", priority: 5},
        { text : "Add some gifs", priority: 10}
    ];
}
```

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:

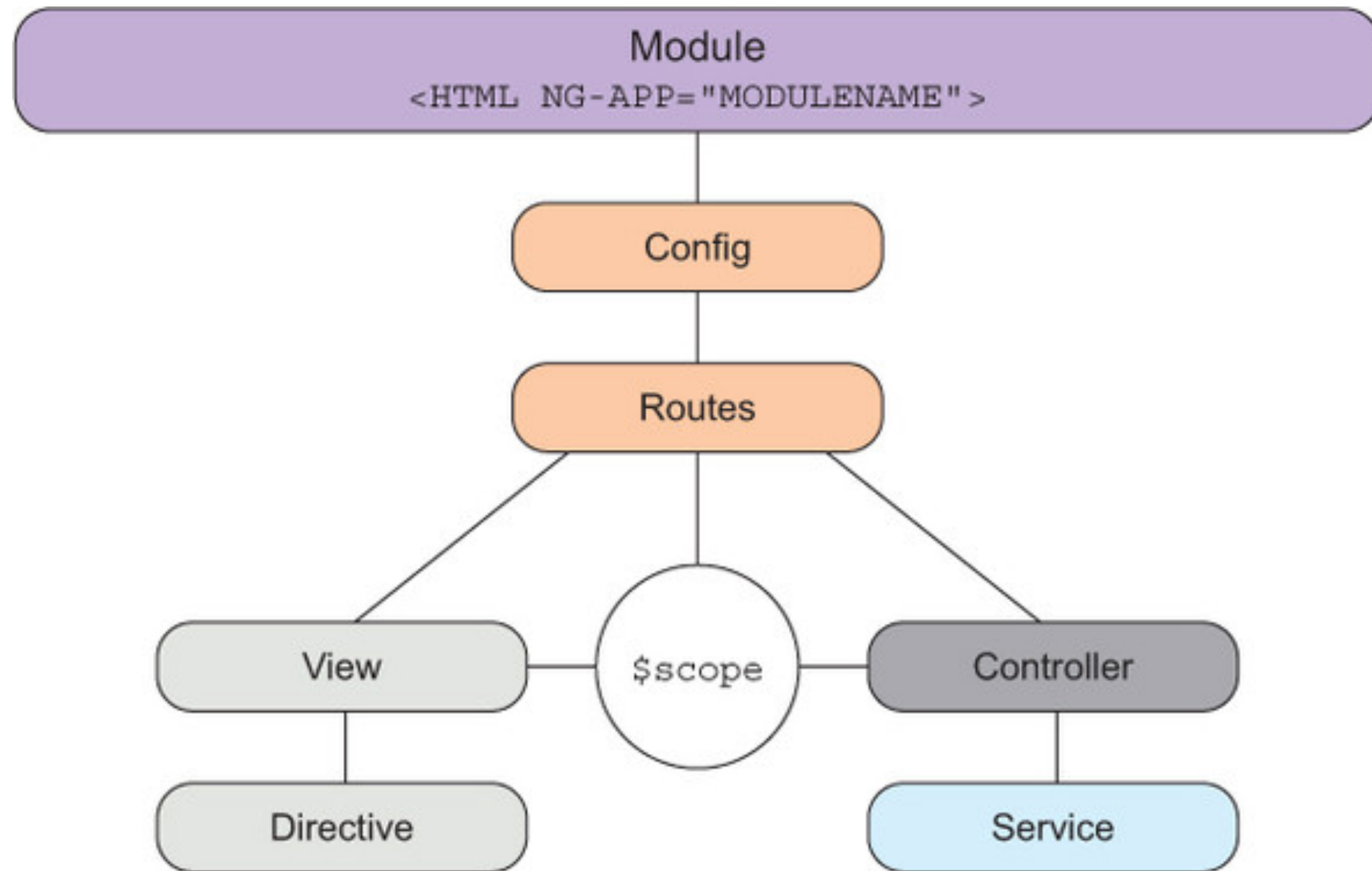
```
var myApp = angular.module('demoApp', []);  
myApp.controller("TodoController", TodoController);
```

- 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">
```

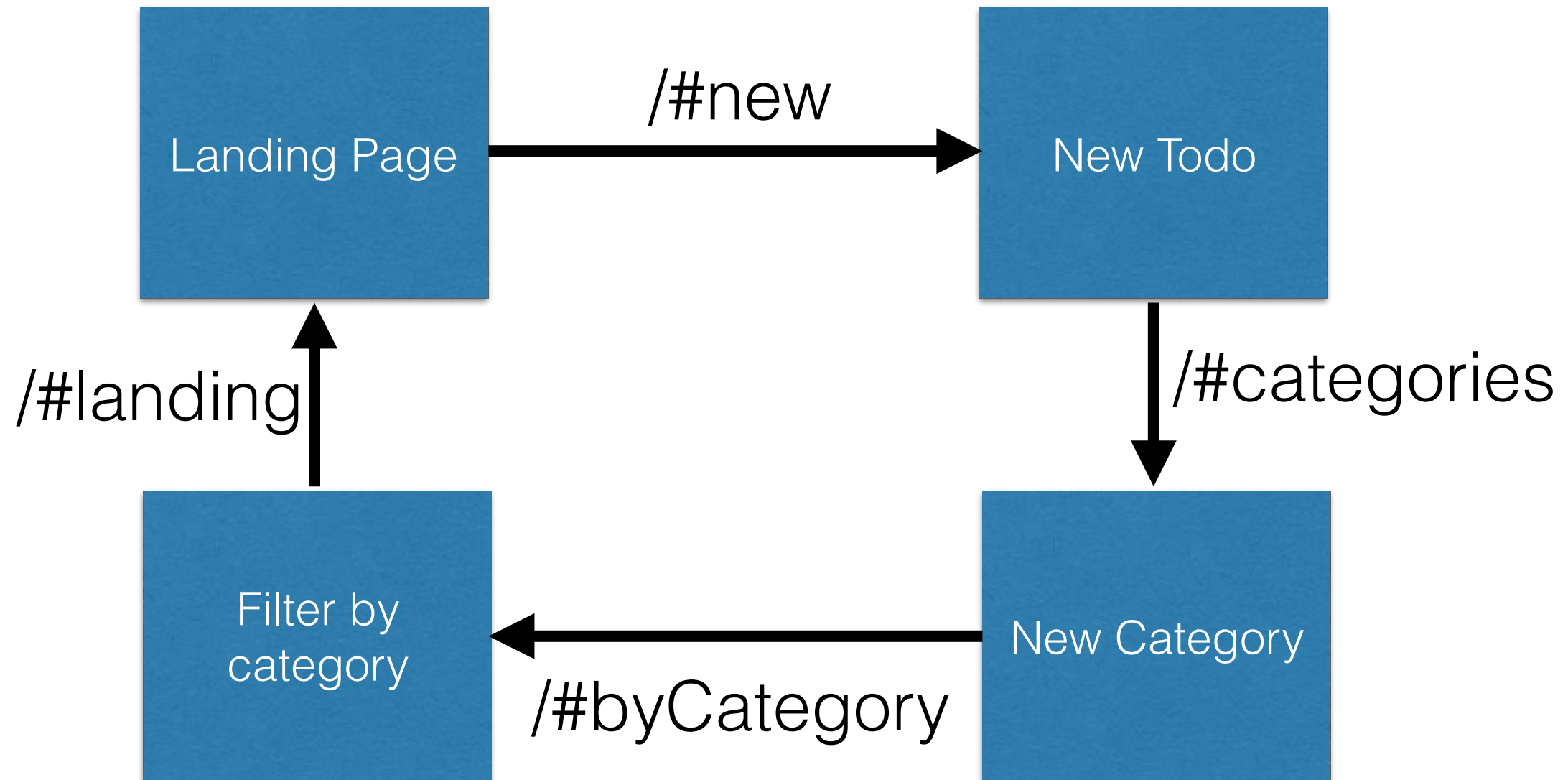
Demo: Modules, Controllers, Firebase

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](mailto:yourname@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?