



# **Web Development**

**COMP 431 / COMP 531**

**Lecture 10: Angular**

**Instructor: Mack Joyner**

**Department of Computer Science, Rice University**

[mjoyner@rice.edu](mailto:mjoyner@rice.edu)

<http://www.clear.rice.edu/comp431>

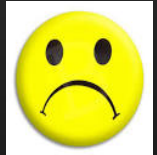
# Recap

- HTML and HTML5, Storage, Canvas
- JavaScript and Scope
- Forms, CSS, Events
- jQuery, AJAX, and fetch
- Modern JS

*Homework Assignment 3*  
*(JS Game)*

Due **TODAY** 9/28

*SVN + Comp 431/531 =*



# Separation of Concerns

In computer science, **separation of concerns** (SoC) is a design principle for **separating** a computer program into distinct sections, such that each section addresses a separate **concern**. A **concern** is a set of information that affects the code of a computer program. -Wikipedia

- Simplify development
- Increase maintainability
- Improve reusability
- Parallelize development
- Promotes Interface development and Encapsulation

# SoC for a Web Application

- Content

HTML

- Presentation

CSS

- Behavior

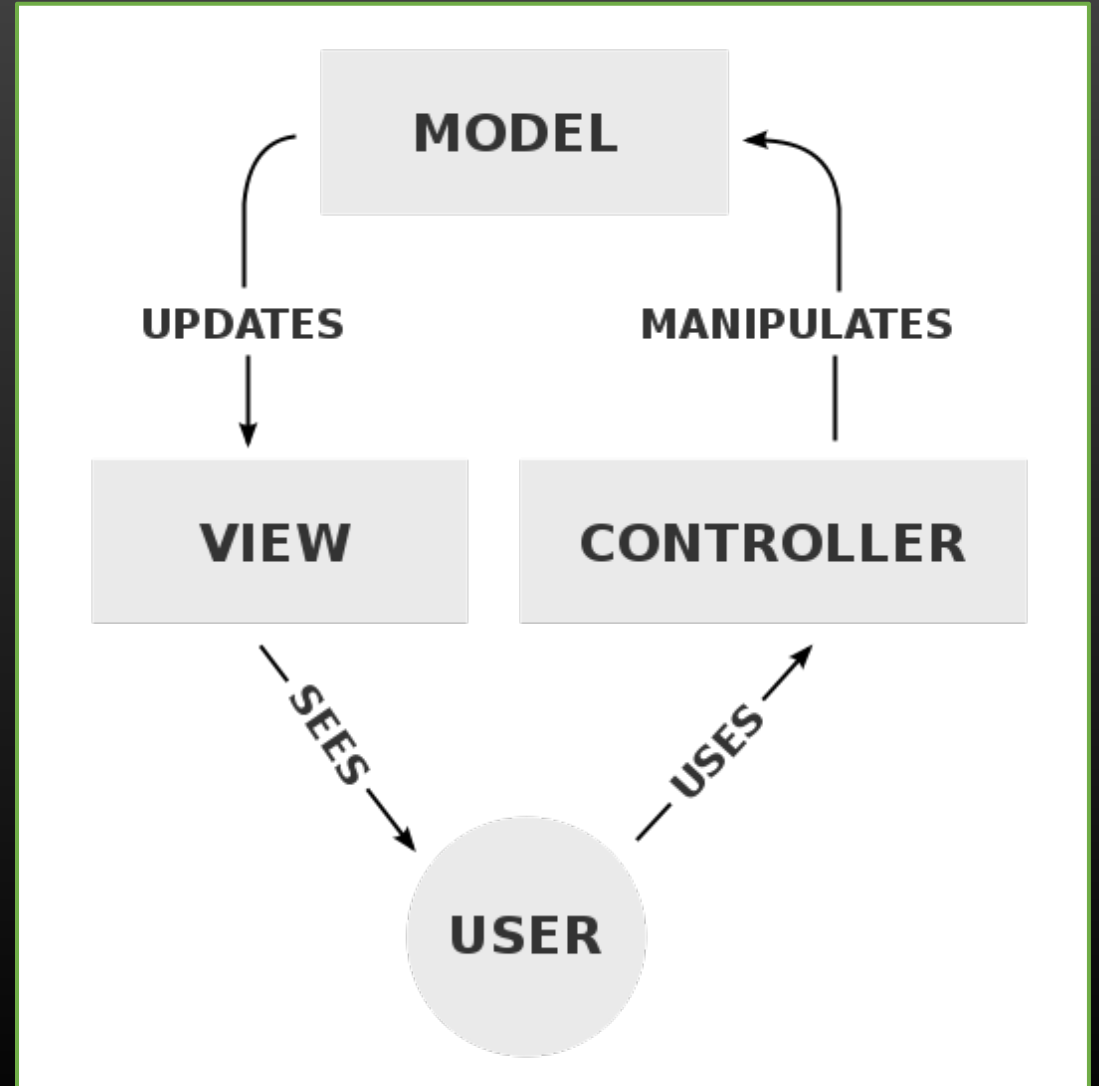
Javascript

## User interface design

- Dynamic layout
- Dynamic content
- Respond to user actions

# Model-View-Controller

- **Model** contains the data
- **View** presents the data to the user in response to the **Model**
- **Controller** sends commands to update the **Model** (change data), or update the **View** (pagination)



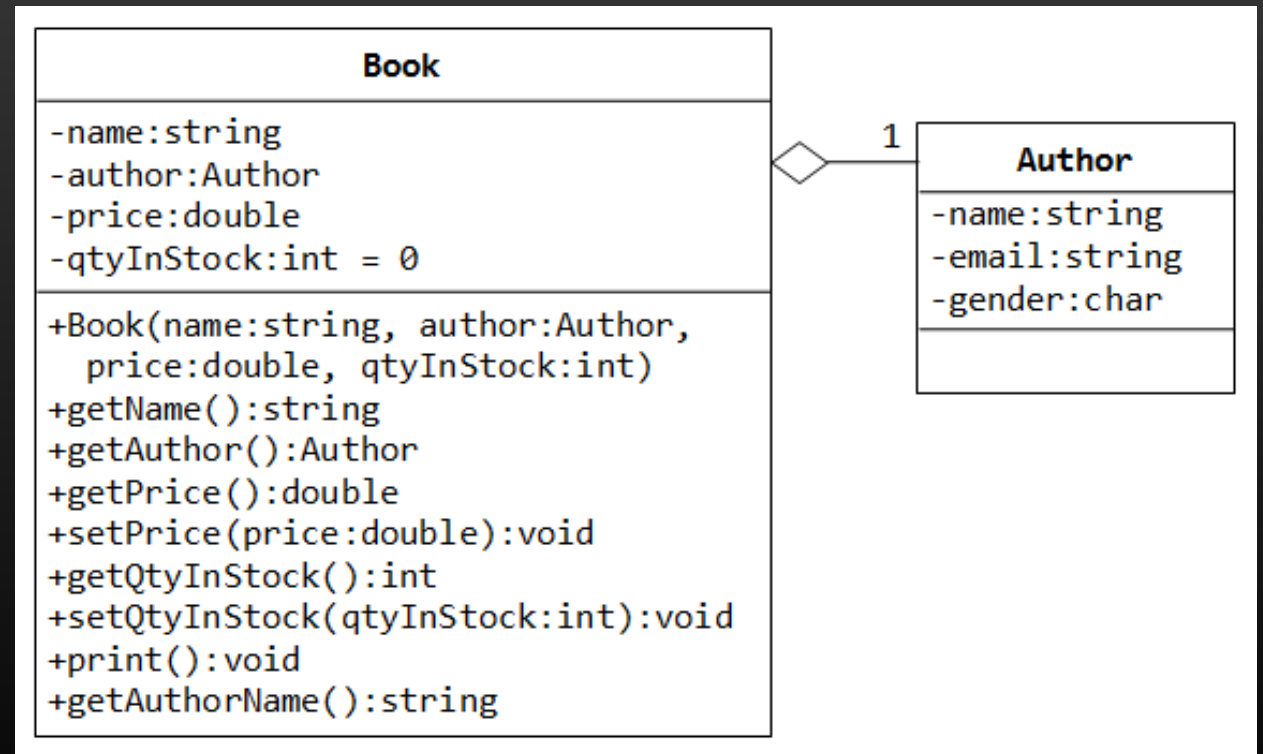
# Example: Online Bookstore

- A database of books
    - Books have properties
  - Display all the books
    - Include selected properties
  - User can select books for purchase
- Model
    - A book
  - View
    - Presentation of books
  - Controller
    - Load model from database
    - Respond to user selection

# Model of a Book

```
Book {  
    author: Author  
    title: string  
    publicationDate: Date  
    publisher: string  
    ISBN: string  
    price: float  
    qtyInStock: int  
}
```

## Classic UML Object Design



# Front-End Frameworks

- Template engines for Views
- Utilize data-binding to populate View from Model / Controller
- Lots to choose from – different styles and techniques



# The Mustache Template System

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <script src="http://cdnjs.cloudflare.com/ajax/libs/mustache.js/0.7.0/mustache.min.js"></script>
5     <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
6 </head>
7 <body>
8
9 <h1>Posts</h1>
10
11 <div id="posts"></div>
12
13 <script id="postTpl" type="text/template">
14 {{#posts}}
15 <h2>{{title}}</h2>
16 <h3>Posted {{date}} by {{author}}</h3>
17 <p>{{body}}</p>
18 <hr>
19 {{/posts}}
20 </script>
```

# The Mustache Template

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <script src="http://cdnjs.cloudflare.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
5   <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
6 </head>
7 <body>
8
9 <h1>Posts</h1>
10
11 <div id="posts"></div>
12
13 <script id="postTpl" type="text/template">
14   {{#posts}}
15   <h2>{{title}}</h2>
16   <h3>Posted {{date}} by {{author}}</h3>
17   <p>{{body}}</p>
18   <hr>
19   {{/posts}}
20 </script>
```

## Posts

### Church-key irony meggings ...

Posted Wed Jul 01 2015 06:40:47 GMT-0500 (Central Daylight Time) by Hipster

Church-key irony meggings biodiesel chillwave bespoke. Excepteur adipisicing polaroid, ex ullamco delectus tilde do nostrud salvia Portland mlkshk sunt culpa. Meditation Tumblr Etsy, delectus sint aute meh chambray ex selfies. Consequat ad disrupt sed. Godard aliqua sed DIY eu, freegan squid art party. Craft beer flannel bitters before they sold out butcher, vegan tousled Godard. Nulla culpa flexitarian, butcher irony pickled polaroid forage 3 wolf moon mixtape hella anim placeat odio paleo.

### 8-bit aesthetic American ...

Posted Wed May 06 2015 12:07:44 GMT-0500 (Central Daylight Time) by Hipster

8-bit aesthetic American Apparel do pork belly. Fap Banksy roof party XOXO reprehenderit, officia cred organic Odd Future 8-bit biodiesel Brooklyn. Marfa distillery placeat Shoreditch et, bicycle rights eiusmod banjo retro. Master cleanse semiotics craft beer occaecat Banksy. Fugiat vegan cray, authentic kitsch banjo Banksy hashtag narwhal bespoke Marfa tilde iPhone. Sustainable plaid meditation, listicle fugiat selvage aesthetic ugh keffiyeh tilde health goth Godard artisan 8-bit. Mumblecore tofu cupidatat, deserunt skateboard sed health goth non farm-to-table Banksy sartorial Godard Etsy distillery.

### Single-origin coffee post-ironic ...

Posted Sat Jul 18 2015 23:32:56 GMT-0500 (Central Daylight Time) by Hipster

Single-origin coffee post-ironic fap messenger bag. Organic High Life nulla viral, elit gentrify Kickstarter Blue Bottle kogi Carles. Post-ironic craft beer aesthetic, Williamsburg gastropub Godard street art occaecat letterpress placeat raw denim. Exercitation Kickstarter quinoa semiotics, listicle trust fund vero readymade kitsch cornhole taxidermy single-origin coffee salvia fap. Trust fund readymade blog pariatur, sriracha wayfarers minim street art slow-carb Carles occupy quis. Neutra in meditation, sint officia irony gastropub. Velit kitsch skateboard tousled, butcher chambray ea +1 Blue Bottle asymmetrical Tumblr cliché.

# The Template System

```
<script id="postTpl" type="text/template">
{{#posts}}
<h2>{{title}}</h2>
<h3>Posted {{date}} by {{author}}</h3>
<p>{{body}}</p>
<hr>
{{/posts}}
</script>
```

```
<script>
window.onload = function() {
  ...

  $.get(url, function(result) {
    var posts = []
    result.text.split('<p>')
      .filter(function(t) { return t.length > 0 })
      .forEach(function(text) {
        var body = text.replace('</p>', '')
        var s = text.split(' ')
        var title = [s[0], s[1], s[2], '...'].join(' ')
        var entry = {
          "author": "Hipster",
          "title": title,
          "body": body,
          "date": new Date(1430012345678 + Math.random()*130e8)
        }
        posts.push(entry)
      })
    var template = $('#postTpl').html()
    var html = Mustache.to_html(template, {"posts":posts});
    $('#posts').html(html)
  })
}
</script>
```

# Knockout JS

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <script src="https://cdnjs.cloudflare.com/ajax/libs/knockout/3.3.0/knockout-min.js"></script>
5   <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
6 </head>
7 <body>
8
9 <h1>Posts</h1>
10
11 <div id="posts" data-bind="foreach: posts">
12   <h2 data-bind="text: title"></h2>
13   <h3>Posted <span data-bind="text: date"></span>
14     by <span data-bind="text: author"></span></h3>
15   <p data-bind="text: body"></p>
16   <hr>
17 </div>
```

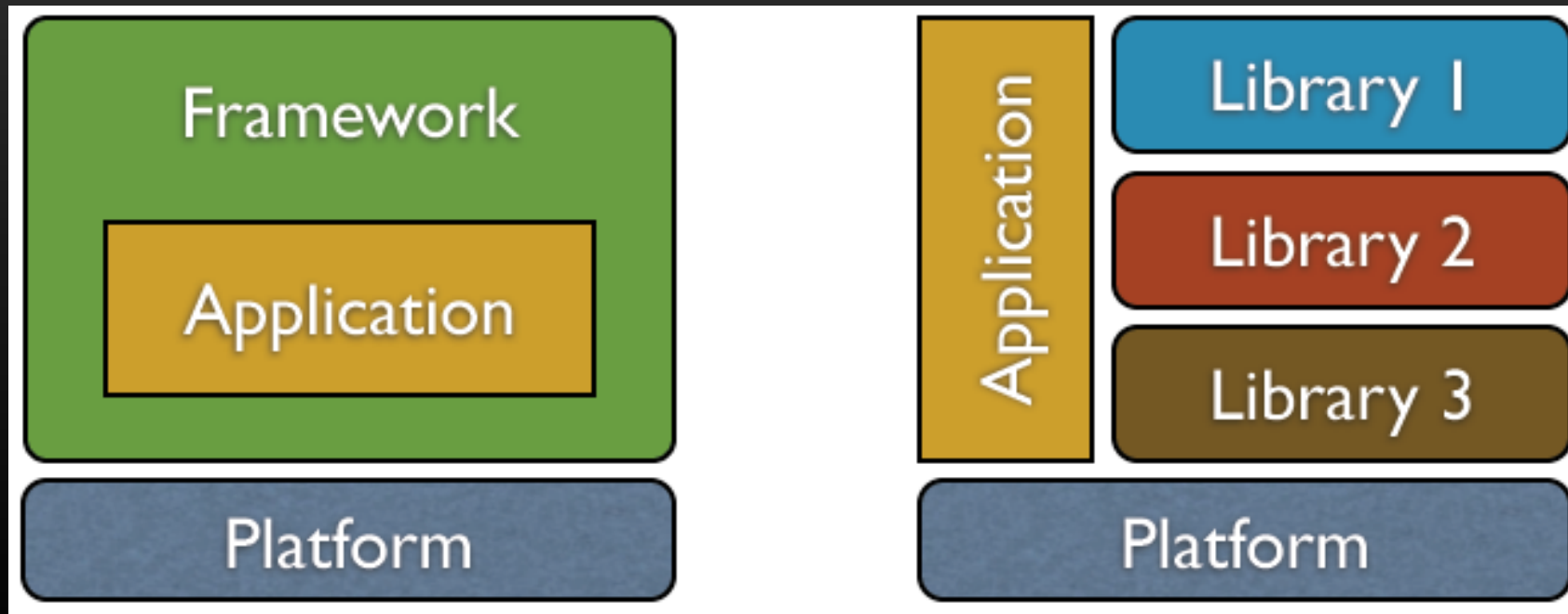
```
<script id="postTpl" type="text/template">
  {{#posts}}
  <h2>{{title}}</h2>
  <h3>Posted {{date}} by {{author}}</h3>
  <p>{{body}}</p>
  <hr>
  {{/posts}}
</script>
```

Mustache Template

# Front-End JavaScript Frameworks

- Mustache, Handlebars, Knockout
- Dojo, MooTools, Backbone
- Ember, React, Angular\*

~Library  
FRAMEWORK



*\* React is more like a library than a framework*

# AngularJS, ReactJS, BackboneJS, and EmberJS Job Trends

AngularJS ×

ReactJS ×

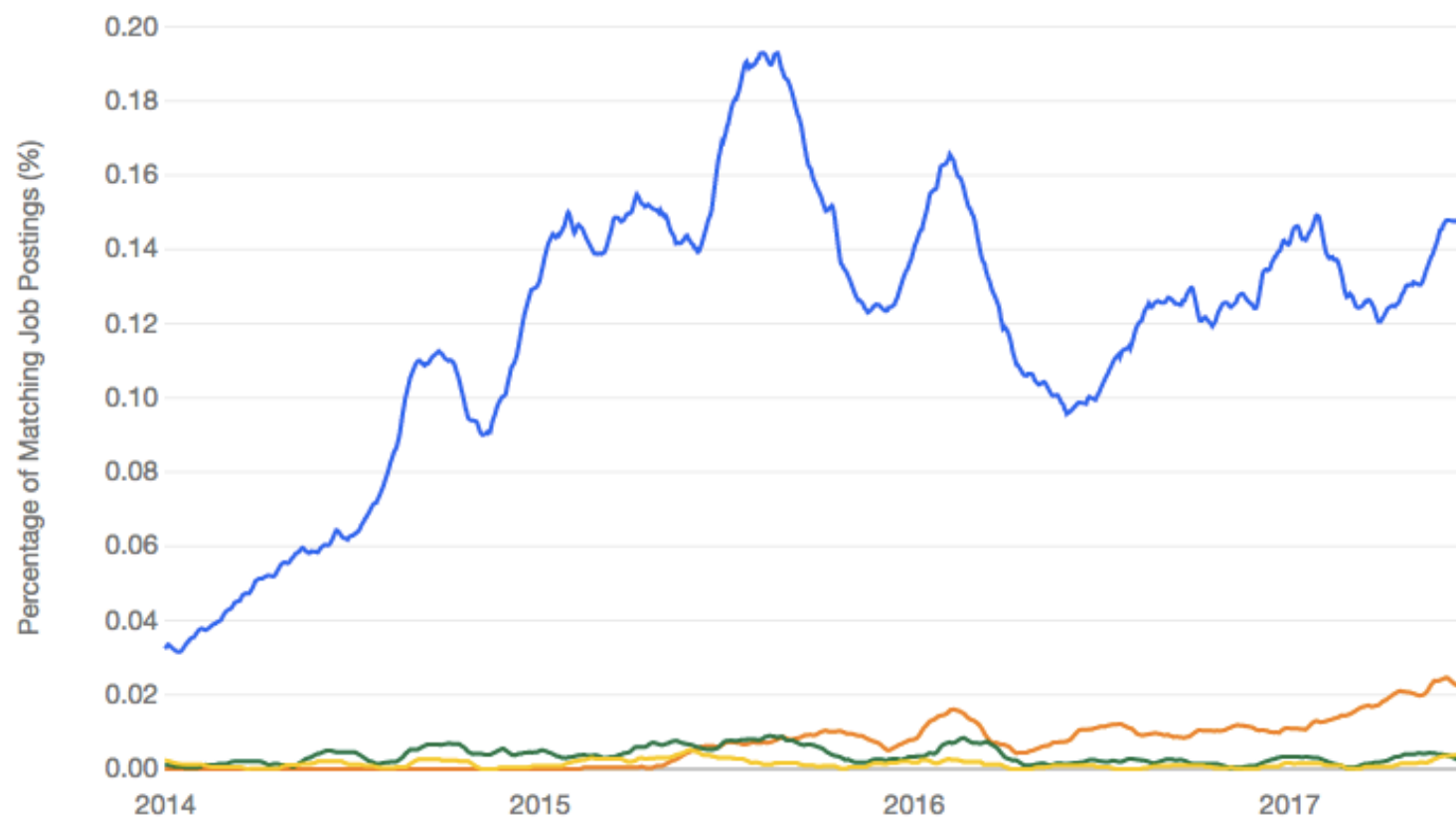
BackboneJS ×

EmberJS ×

+ Add Term

Find Trends

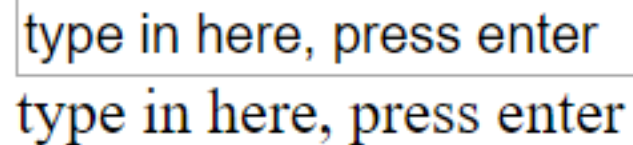
## Job Postings



# Write our own Framework

- To have a better understanding of how template engines work
- Consider a simple page:

```
<body>  
  <input data-model="message" type="text" />  
  <div data-model="message"></div>  
  <script src="app.js"></script>  
</body>
```



type in here, press enter  
type in here, press enter

- The idea:
  1. We have a map from model source to target(s)
  2. On page load, register elements as sources or targets
  3. Sources have event handlers that update the targets

<https://www.clear.rice.edu/comp431/sample/framework>

<https://www.clear.rice.edu/comp431/sample/framework/app.js>

# Dynamic elements are more challenging

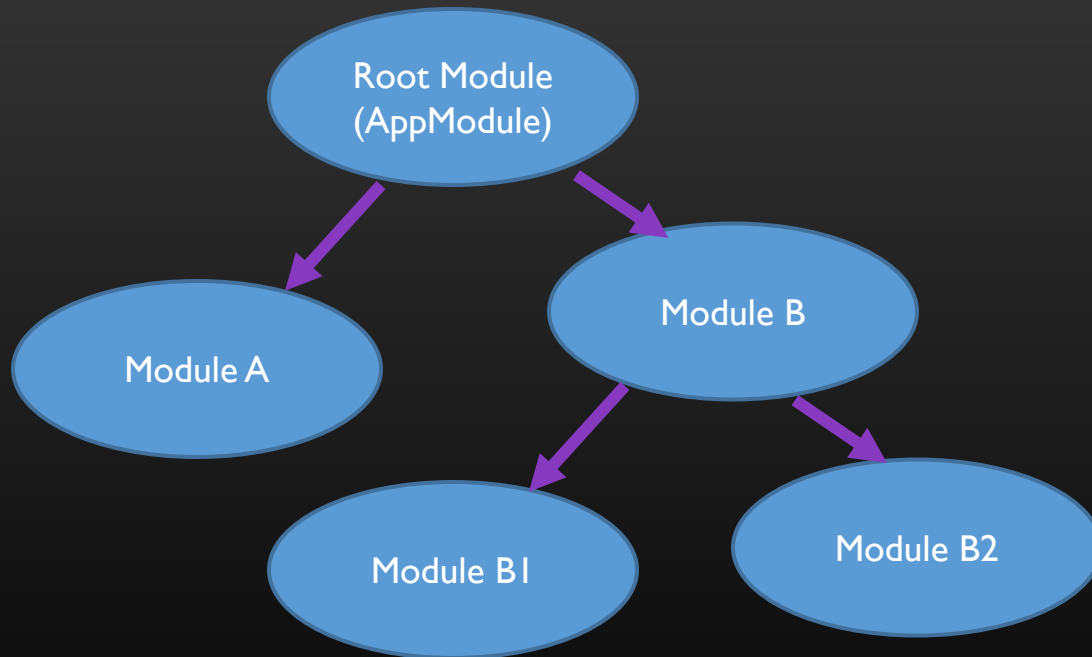
```
<tr data-repeat="book in books">  
  <td>{{ book.author }}</td>  
  <td>{{ book.title }}</td>  
  <td>{{ book.price }}</td>  
</tr>
```





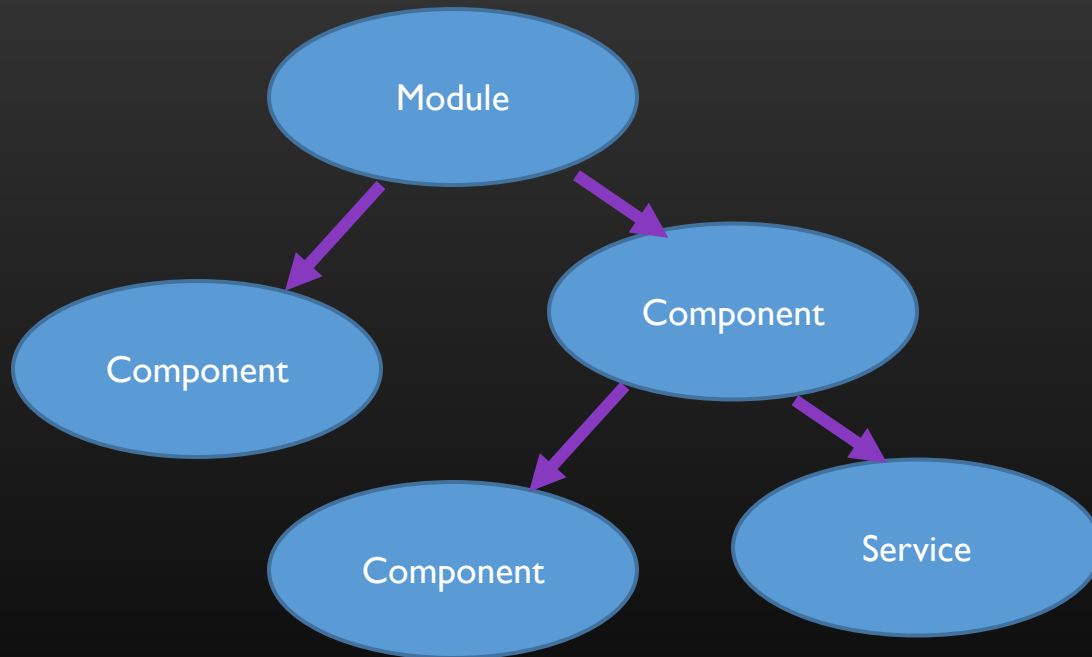
- Angular is a structural framework for dynamic web apps
- AngularJS first released in June 2012
  - Designed by Misko Hevery
- Angular
  - Framework completely rewritten
  - Component-based with templates
  - Mobile support

# Angular Modular Architecture



- Modular
- Component-based, managed by modules
- Templates
- Services

# Angular Modular Architecture



- May divide screen into multiple views
- Each view may be controlled by a separate component
- Components can communicate with other components

# Starting the Angular Application

```
main.ts
1  import { enableProdMode } from '@angular/core';
2  import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4  import { AppModule } from './app/app.module';
5  import { environment } from './environments/environment';
6
7  if (environment.production) {
8    enableProdMode();
9  }
10
11  platformBrowserDynamic().bootstrapModule(AppModule)
12    .catch(err => console.log(err));
13
```

# Angular Root Module

## Module Declarator

## Components in Module

## Adds bootstrapping (execution)

```
app.module.ts
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppComponent } from './app.component';
5
6  @NgModule({
7    declarations: [
8      AppComponent
9    ],
10   imports: [
11     BrowserModule,
12
13   ],
14   providers: [],
15   bootstrap: [AppComponent]
16 })
17 export class AppModule { }
```

# Angular Root App Component

Component declarator

Create component tag

Place template in separate file

Prepare root app component

```
app.component.ts
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'app';
10 }
```

# SoC for Templates

```
app.component.html
1  <!--The content below is only a placeholder and can be replaced.-->
2  <div style="text-align:center">
3    <h1>
4      Welcome to {{title}}!
5    </h1>
6    
9  </div>
10 <h2>Here are some links to help you start: </h2>
11 <ul>
12   <li>
13     <a target="_blank" rel="noopener" href="https://angular.io/tutorial">Tour
14   </li>
15   <li>
16     <a target="_blank" rel="noopener" href="https://github.com/angular/angular
17   </li>
18   <li>
19     <a target="_blank" rel="noopener" href="https://blog.angular.io/">Angular
20   </li>
21 </ul>
```

Template used by root component

# Angular App index.html File

```
index.html
1  <!doctype html>
2  <html lang="en">
3  <head>
4    <meta charset="utf-8">
5    <title>HelloWorld</title>
6    <base href="/">
7
8    <meta name="viewport" content="width=device-width, initial-scale=1">
9    <link rel="icon" type="image/x-icon" href="favicon.ico">
10 </head>
11 <body>
12   <app-root></app-root>
13 </body>
14 </html>
15 |
```

Tag defined in root component selector



# Moving to Git

- Homework 3 will still use svn
- Homework 4 and on... git
- Inclass exercise 10 and on... git
- Create a bitbucket account at <https://bitbucket.org>
  - Repos will be private (cost=free)
  - Use your netid rice email when creating an account (free to academic plan)
  - Create a repo and name it comp431\_531 (commit the README)
  - Invite an instructor ([mjoyner@rice.edu](mailto:mjoyner@rice.edu)) to this repo
  - Clone a local copy of the repo:  
git clone https://netid@bitbucket.org/netid/comp431\_531.git