RICE®

COMP 431 / COMP 531

Model-View-Controller (and others)

Scott E Pollack, PhD

February 11, 2016

Recap

- HTML and HTML5
- JavaScript and JS Libraries
- Forms
- CSS and Style Frameworks
- Events
- Homework Assignment 4 (JavaScript Game)
  - **Due Thursday 2/18**

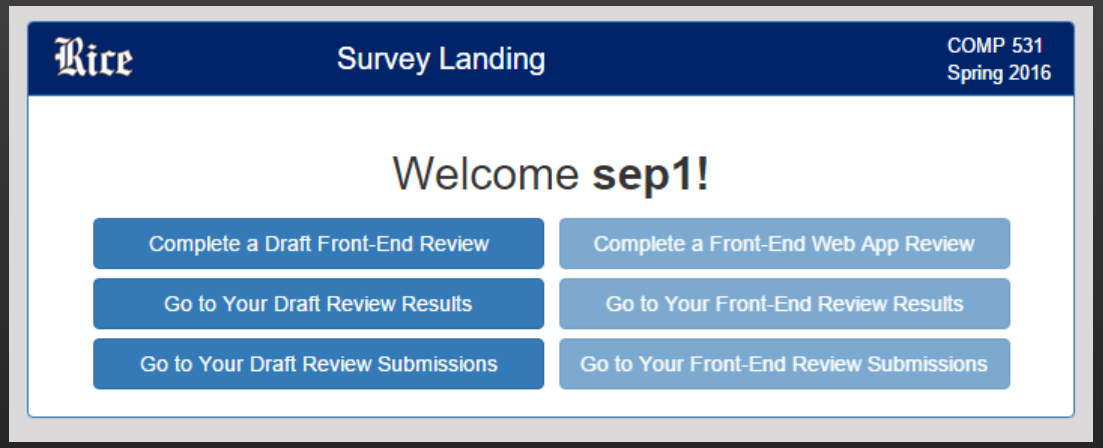
~~Style (Bootstrap)~~
~~Libraries (jQuery)~~
MVC
AngularJS

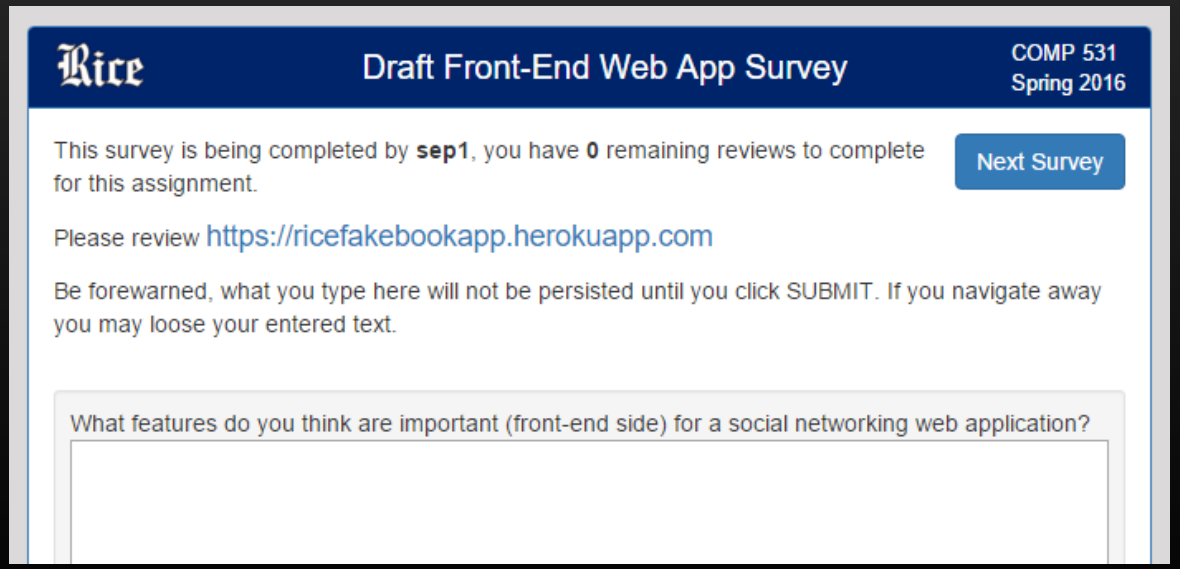
COMP 531
Draft Front-End Review
Due Tuesday 2/23

http://webdev-dummy.herokuapp.com/survey

COMP 531 Draft Front-End Review

- Go to this address and log in using your netid and 3-word password supplied to you by email for the dummy server
- You will be asked to review 7 websites of your peers
- Provide useful, constructive feedback
- Provide a critical comparative evaluation of each site with yours





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

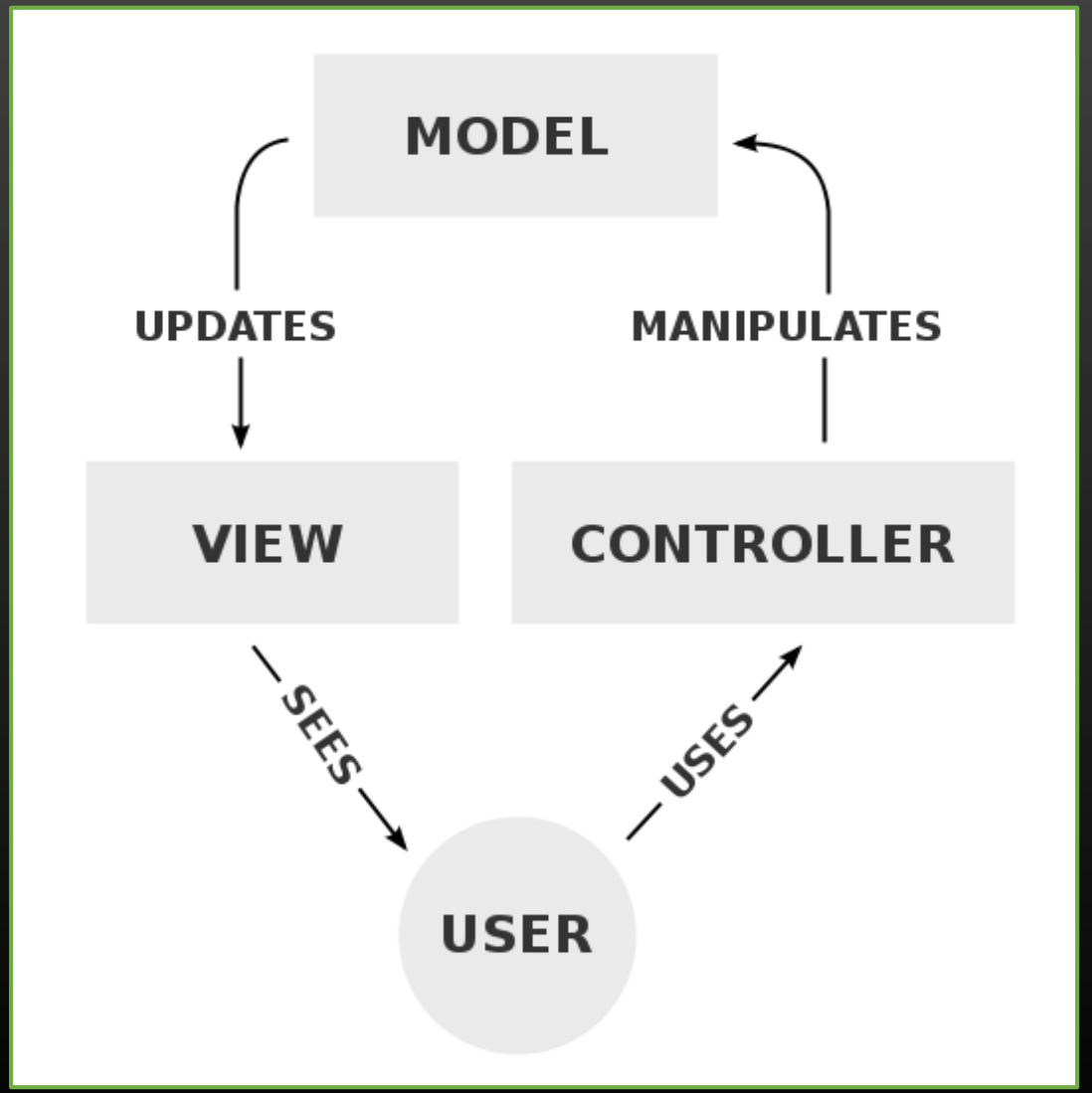
JAVASCRIPT

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

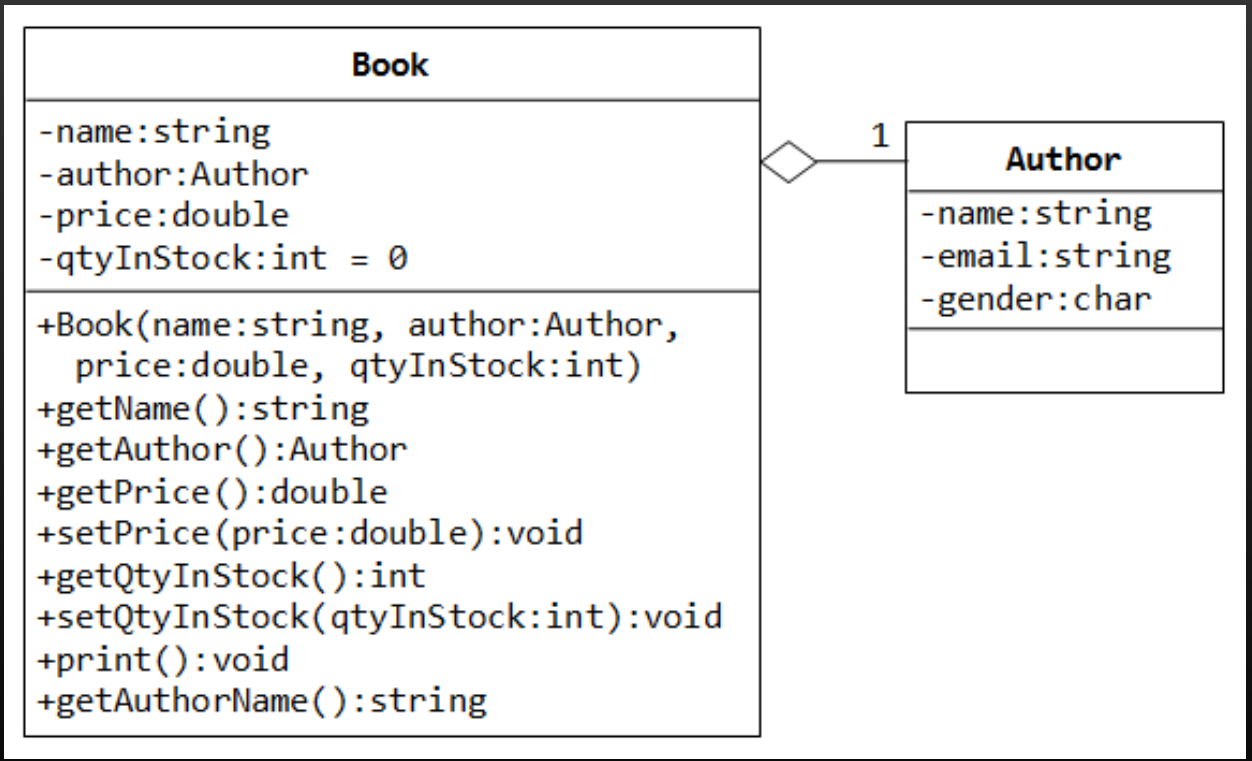
- A database of books
  - Books have properties
- Display all the books
  - Include selected properties
- User can select books for purchase

- Model
  - A book
- View
  - Presentation of books
- Controller
  - Load model from database
  - Respond to user selection

Model of a Book

```
Book {
    author: string
    title: string
    publicationDate: date
    publisher: string
    ISBN: string
    price: float
}
```

Classic UML Object Design




WARNING
php

“Dynamic” View of Books

```
<table>
    <tr>
        <th>Author</th><th>Title</th><th>Price</th>
    </tr>

<?php foreach ($books as $book): ?>
    <tr>
        <td> <?php echo $book['author']; ?> </td>
        <td> <?php echo $book['title']; ?> </td>
        <td> <?php echo $book['price']; ?> </td>
    </tr>
<?php endforeach; ?>

</table>
```

Book Controller

```php
class BookController {
    public function init() {
        $this->Model = new BookModel();

        // $books is in "scope" for the view
        $this->$books = $this->Model->getAll();
    }
}

new BookController().init();
```

```php
class Book {
    public $author;
    public $title;
    ...
    // create a Book from a database row
    public function __construct($row) {
        $this->$author = ...
    }
}
```

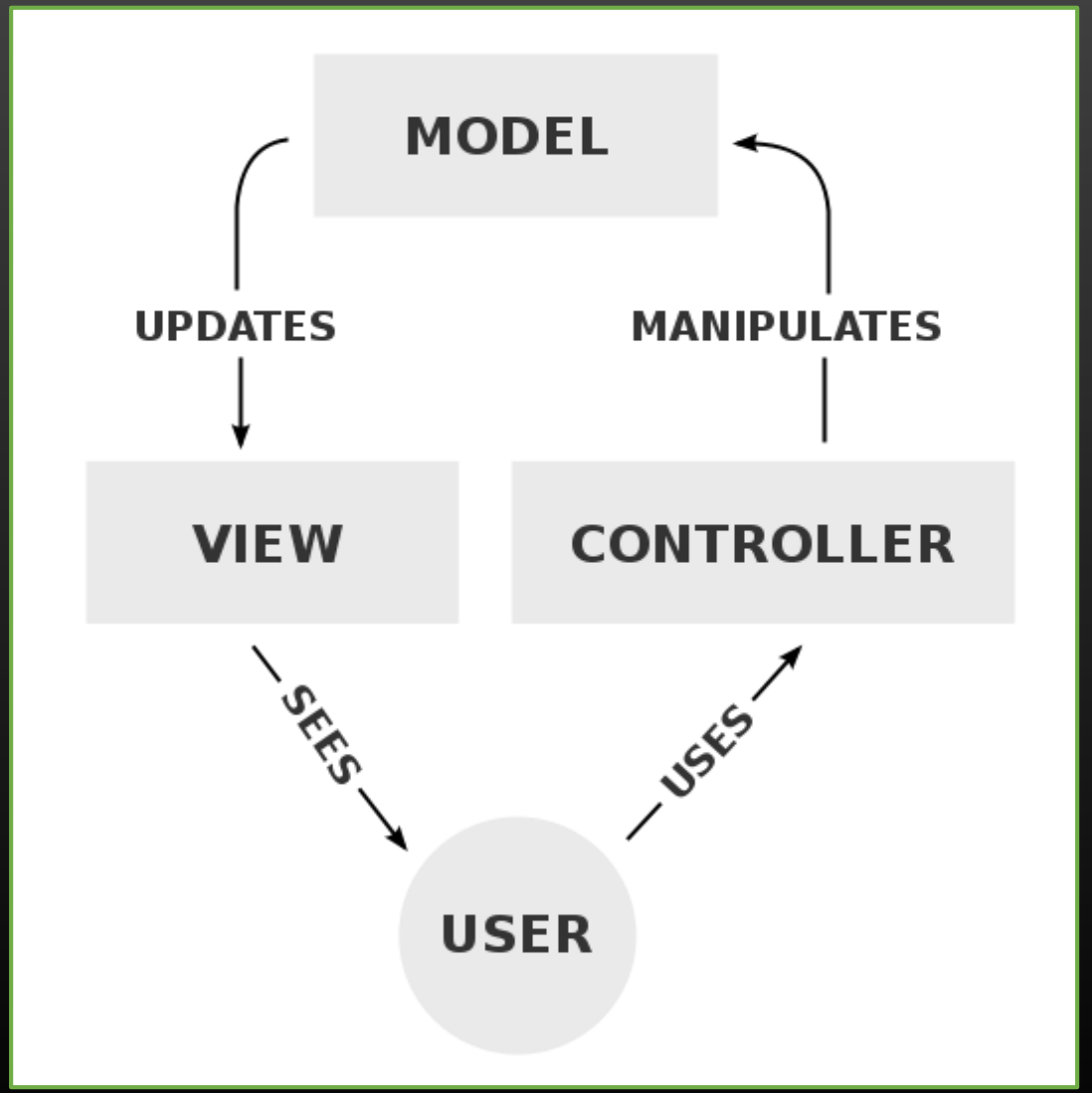
DTO

```php
class BookModel {
    // make database query to get all of the books.
    // return the ResultSet as an array of Books
    public function getAll() {
        $results = array();
        $resultSet = $this->db->getAllBooks());
        foreach ($resultSet as $row) {
            $results[] = new Book($row);
        }
        return $results;
    }
}
```

DAO

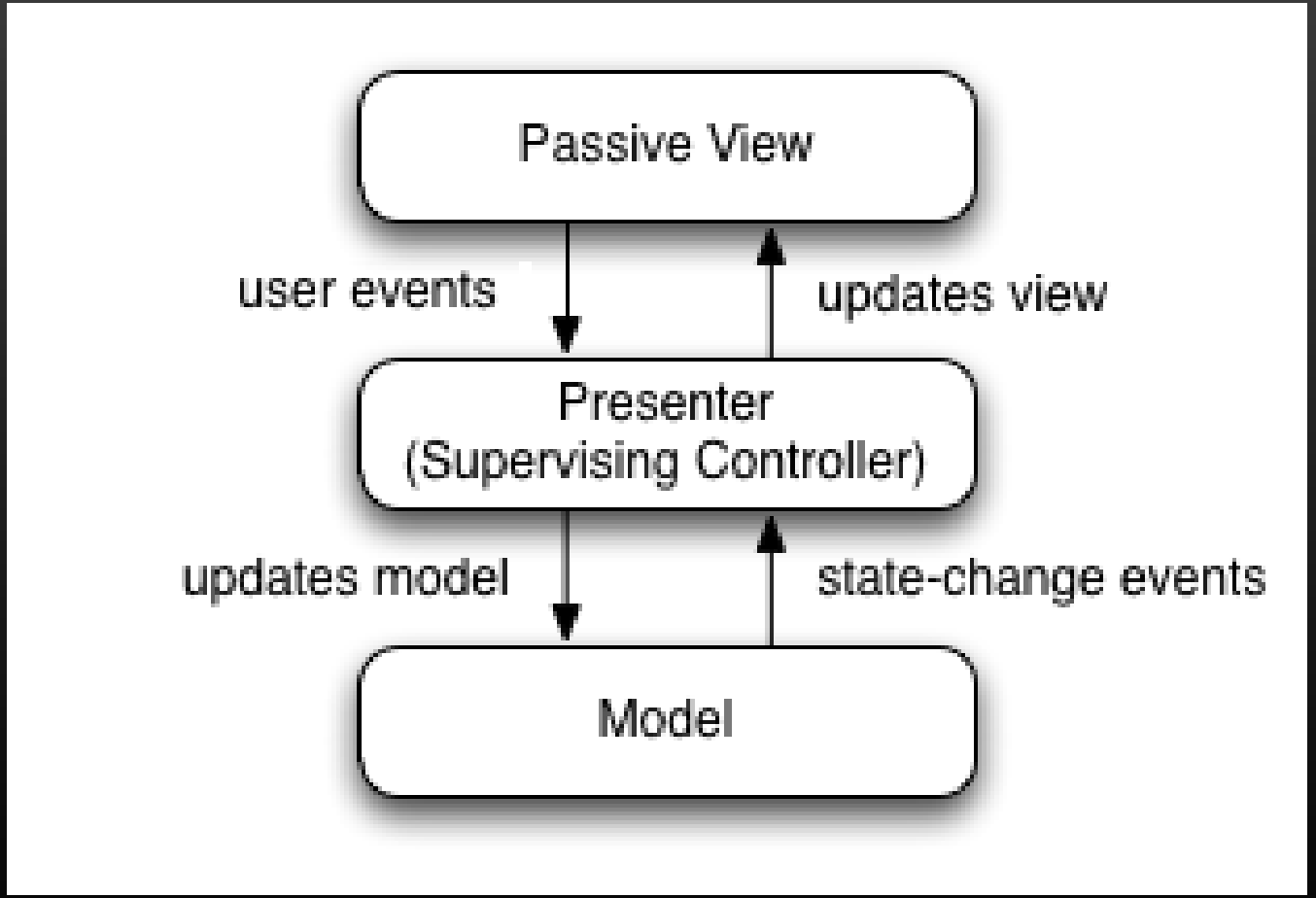
Model-View-Controller

- 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)

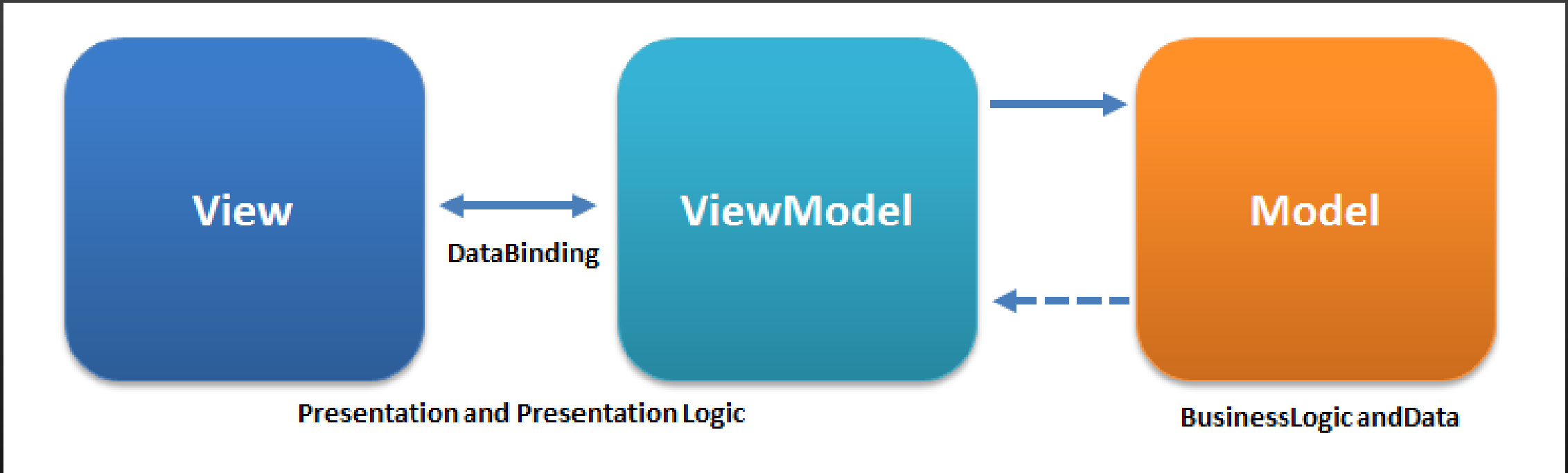


Model-View-Presenter

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



Model-View-ViewModel



- ViewModel is a state representation of the data from the Model
- Binder utilizes declarative data and command binding in the markup for data synchronization

MVVM vs MVC

MVVM

1. ViewModel encapsulates presentation logic and state.
2. ViewModel is an optional Pattern.
3. User hits the View first
4. View can't see the Model
5. View has an instance of the ViewModel

MVC

1. The Controller determine the Application Flow.
2. Controller is a must.
3. User hits the Controller first
4. The View knows about the Model
5. View gets an instance of the Model

https://khatrishashank.wordpress.com/2013/10/11/a-comparison-of-architectural-patternsmvvm-and-mvc/

Front-End Frameworks

- Template engines for Views
- Utilize data-binding to populate View from ViewModel / Controller / Presenter
- Lots to choose from – different styles and techniques
- Some modular, others opinionated

View Template

Preprocessed on Server

```
<table>
    <tr>
        <th>Author</th><th>Title</th><th>Price</th>
    </tr>

<?php foreach ($books as $book): ?>
    <tr>
        <td> <?php echo $book['author']; ?> </td>
        <td> <?php echo $book['title']; ?> </td>
        <td> <?php echo $book['price']; ?> </td>
    </tr>
<?php endforeach; ?>

</table>
```

Front-End Controlled by JS

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

The Mustache Template System

```
<!DOCTYPE html>
<html>
<head>
    <script src="http://cdnjs.cloudflare.com/ajax/libs/mustache.js/0.7.0/mustache.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
</head>
<body>

<h1>Posts</h1>

<div id="posts"></div>

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

The Mustache Templ

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <script src="http://cdnjs.cloudflare.co
5      <script src="https://ajax.googleapis.co
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 cliche.

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() {
    var url = 'http://hipsterjesus.com/api/'

    $.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(14300012345678 + Math.random()*130e8)
                }
                posts.push(entry)
            })
        var template = $('#postTpl').html()
        var html = Mustache.to_html(template, {"posts":posts});
        $('#posts').html(html)
    })
}
</script>
```

Knockout JS

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/knockout/3.3.0/knockout-min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
</head>
<body>

<h1>Posts</h1>

<div id="posts" data-bind="foreach: posts">
    <h2 data-bind="text: title"></h2>
    <h3>Posted <span data-bind="text: date"></span>
        by <span data-bind="text: author"</span></h3>
    <p data-bind="text: body"></p>
    <hr>
</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

Knockout JS

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://cdnjs.cloudflare.com/ajax/
    <script src="https://ajax.googleapis.com/ajax/l
</head>
<body>

<h1>Posts</h1>

<div id="posts" data-bind="foreach: posts">
    <h2 data-bind="text: title"></h2>
    <h3>Posted <span data-bind="text: date"></span>
        by <span data-bind="text: author"</span></h
    <p data-bind="text: body"></p>
    <hr>
</div>
```

```
var template = $('#postTpl').html()
var html = Mustache.to_html(template, {"posts":posts});
$('#posts').html(html)
```

Mustache

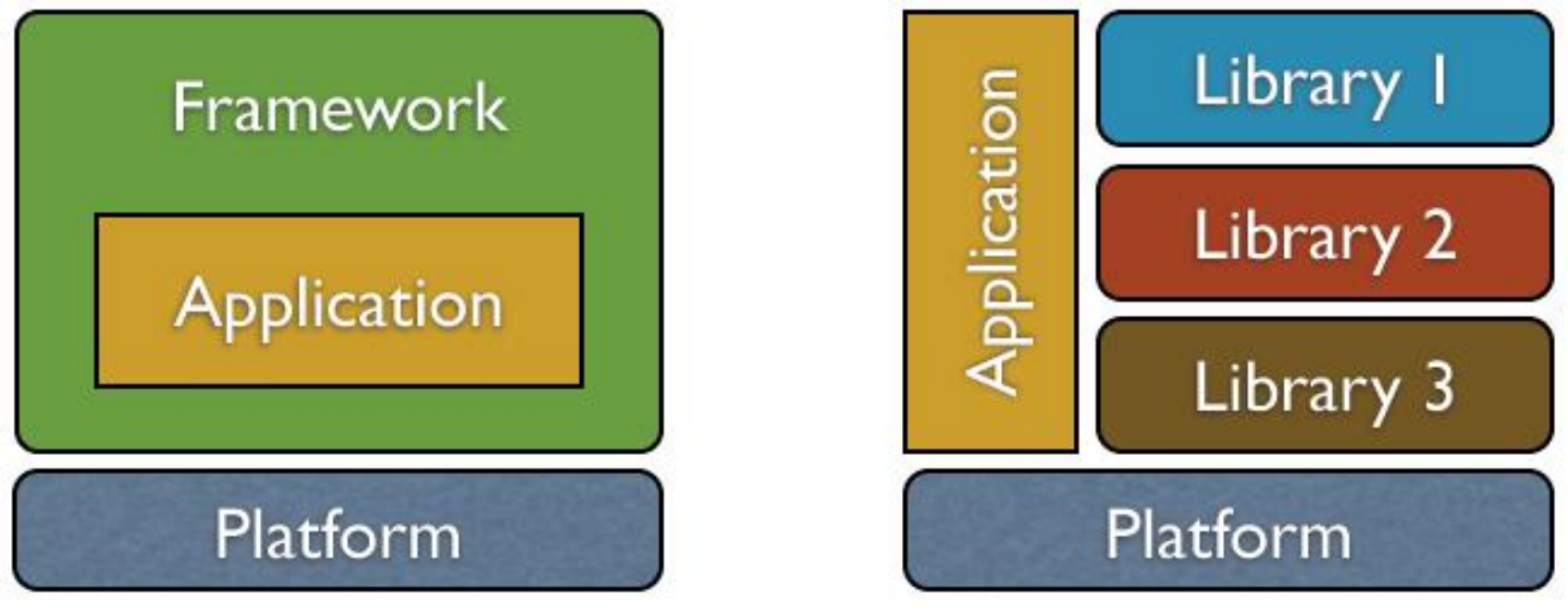
```
function PostModel(text) {
    var body = text.replace('</p>', '')
    var s = text.split(' ')
    var title = [s[0], s[1], s[2], '...'].join(' ')

    this.author = "Hipster"
    this.title = title
    this.body = body
    this.date = new Date(1430012345678 + Math.random()*130e8)
}

function PostViewModel() {
    var vm = this;
    vm.posts = ko.observableArray([])

    var url = 'http://hipsterjesus.com/api/'
    $.get(url, function(result) {
        result.text.split('<p>')
            .filter(function(t) { return t.length > 0 })
            .forEach(function(text) {
                vm.posts.push(new PostModel(text))
            })
    })
}

window.onload = function() {
    ko.applyBindings(new PostViewModel());
}
```

Front-End JavaScript Frameworks

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

~Library

FRAMEWORK



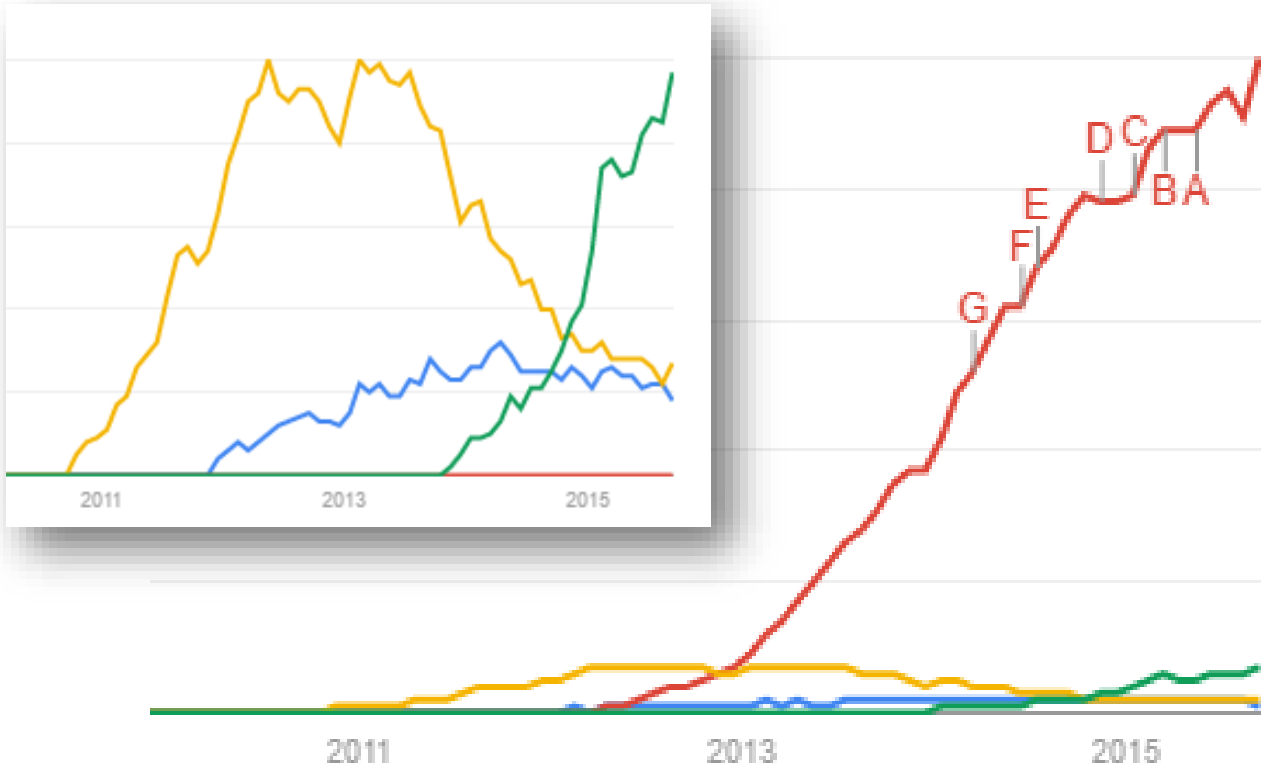
Front-End JavaScript Frameworks

BACKBONE.JS

ANGULARJS by Google

React

ember



emberjs
Search term

angularjs
Search term

backbone.js
Search term

reactjs
Search term

“the superheroic framework”

- Model-View-(Whatever)
- Single-Page-Application
- Two-Way Data Binding
  - Reduces boilerplate
  - Fast updates View-to-Model and back
- Observation built-in
- Separation of Concerns
  - HTML template vs JS logic
- Plain Old Javascript Objects

- Complete front-end stack
- Unit testing with Karma
- Great for tiny apps
- Good for medium apps
- Even works for large apps
  - Be careful of too many watchers
- Opinionated!
  - There is a “way” to do things

See also http://jeffwhelpley.com/angularjs/

Run a Local Web Server using Python

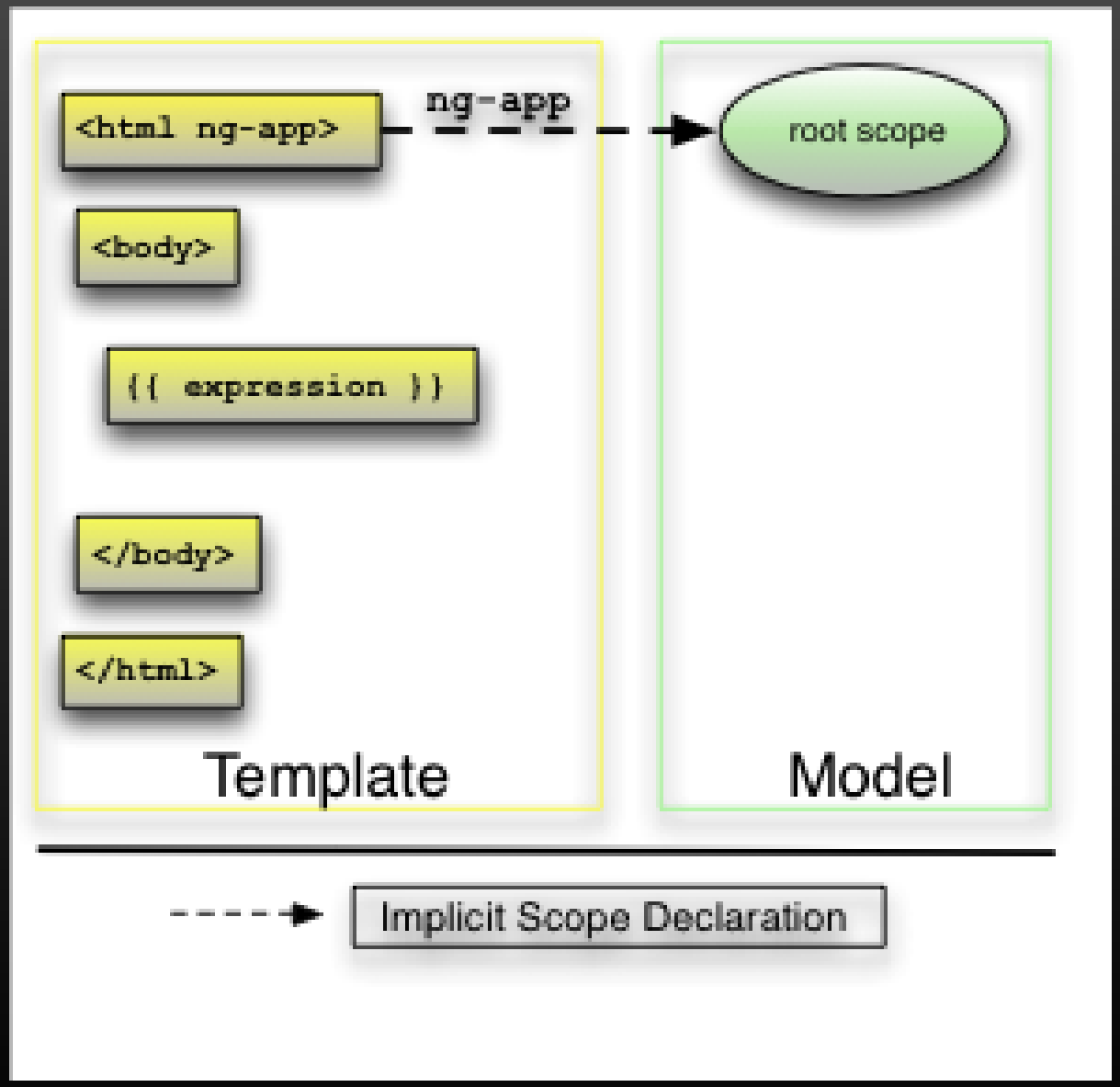
Angular works best when using http(s) *NOT* file:///

```
cd frontend

python -m SimpleHTTPServer 8080

            -OR-

python3 -m http.server 8080
```

Then navigate to http://localhost:8080

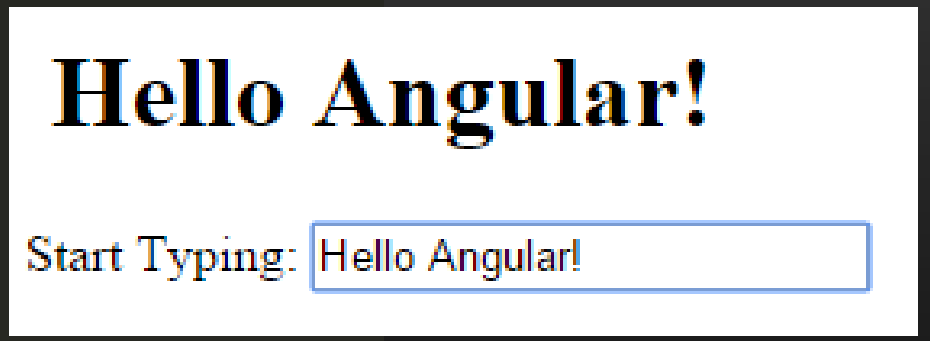
Angular's Structure

- App
  - The ngApp defines the root scope
- View
  - Use handlebar style for templating
- Controller
  - In charge of section of one view
  - Can have multiple Controllers on a single page (SPA)
  - Will have multiple view, therefore multiple Controllers



```
<!DOCTYPE html>
<html ng-app           >
<head>
  <meta charset="utf-8" />
  <title>Hello Angular</title>

  <script src="lib/angular-1.4.6/angular.js"></script>




</head>
<body>

<div>
  <h1> {{ userInput }}</h1>
  <label for="in">Start Typing: </label>
  <input id="in" type="text" ng-model="userInput">
</div>

</body>
</html>
```

Model

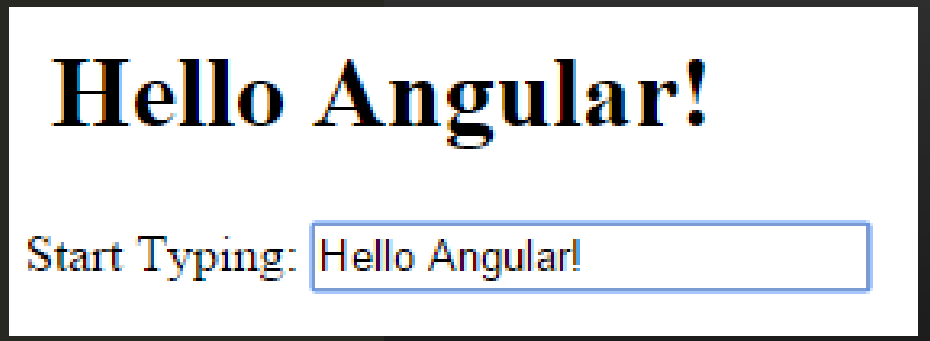


Two-Way Data Binding

```
1  <!DOCTYPE html>
2  <html ng-app="helloNg">
3  <head>
4    <meta charset="utf-8" />
5    <title>Hello Angular</title>
6
7    <script src="lib/angular-1.4.6/angular.js"></script>
8
9    <script>
10   angular.module('helloNg', [])
11   </script>
12
13 </head>
14 <body>
15
16 <div>
17   <h1> {{ userInput }}</h1>
18   <label for="in">Start Typing: </label>
19   <input id="in" type="text" ng-model="userInput">
20 </div>
21
22 </body>
23 </html>
```

ngApp is a Module

Model



Two-Way Data Binding

A bit more interesting…

```
1  <!DOCTYPE html>
2  <html ng-app="helloNg">
3  <head>
4    <meta charset="utf-8" />
5    <title>Hello Angular</title>
6    <script src="lib/angular-1.4.6/angular.js"></script>
7    <script src="hello-angular.js"></script>
8  </head>
9  <body ng-controller="MainCtrl">
10
11 <div ng-repeat="post in posts">
12     <h3>{{post.title}}</h3>
13     <p>{{post.body}}</p>
14 </div>
15
16 </body>
17 </html>
```

the first

message

the second

lorem ipsum

the third

e plurbis unum

```
4  angular.module('helloNg', [])
5      .controller('MainCtrl', MainCtrl);
6
7  MainCtrl.$inject = ['$scope']
8  function MainCtrl($scope) {
9      $scope.posts = [
10         {'title':'the first', 'body':'messa
11         {'title':'the second', 'body':'lore
12         {'title':'the third', 'body':'e plu
13     ]
14 }
```

In-Class Exercise: Hello AngularJS

https://angularjs.org/ *(we want Angular 1 not 2)*

- Add AngularJS to your web app
- Use two-way data binding between the "status headline" and the "text input" for updating it.
- Add a Controller
- Using the Controller and ng-repeat, remove the hardcoded posts from main.html (instead have hard coded data in the main.js file)

***Turnin* main.html&js *to* COMP431-S16:inclass-10**