



CS55.11 JavaScript

Fall 2016 ~ Ethan Wilde

ewilde@santarosa.edu



**SANTA ROSA
JUNIOR COLLEGE**

Course Outline

1 Intro to Javascript, Part 1	10 Data Visualization with D3
2 Intro to Javascript, Part 2	11 Further Explorations with D3
3 <i>no class meeting</i>	12 Browser Games: Phaser
4 Exploring the jQuery Library	13 Further with Phaser
5 Exploring the jQuery UI Library	14 Extending Phaser with JSON
6 Exploring jQuery Plug-ins	15 Single Page Apps: Angular.js
7 Exploring Google Maps API	16 Angular.js
8 AJAX: Working with JSON	17 Server-side JS + Final Review
9 Midterm Review + Midterm	18 Final Exam (no classes)

- Get all of the details in the complete syllabus on Canvas.

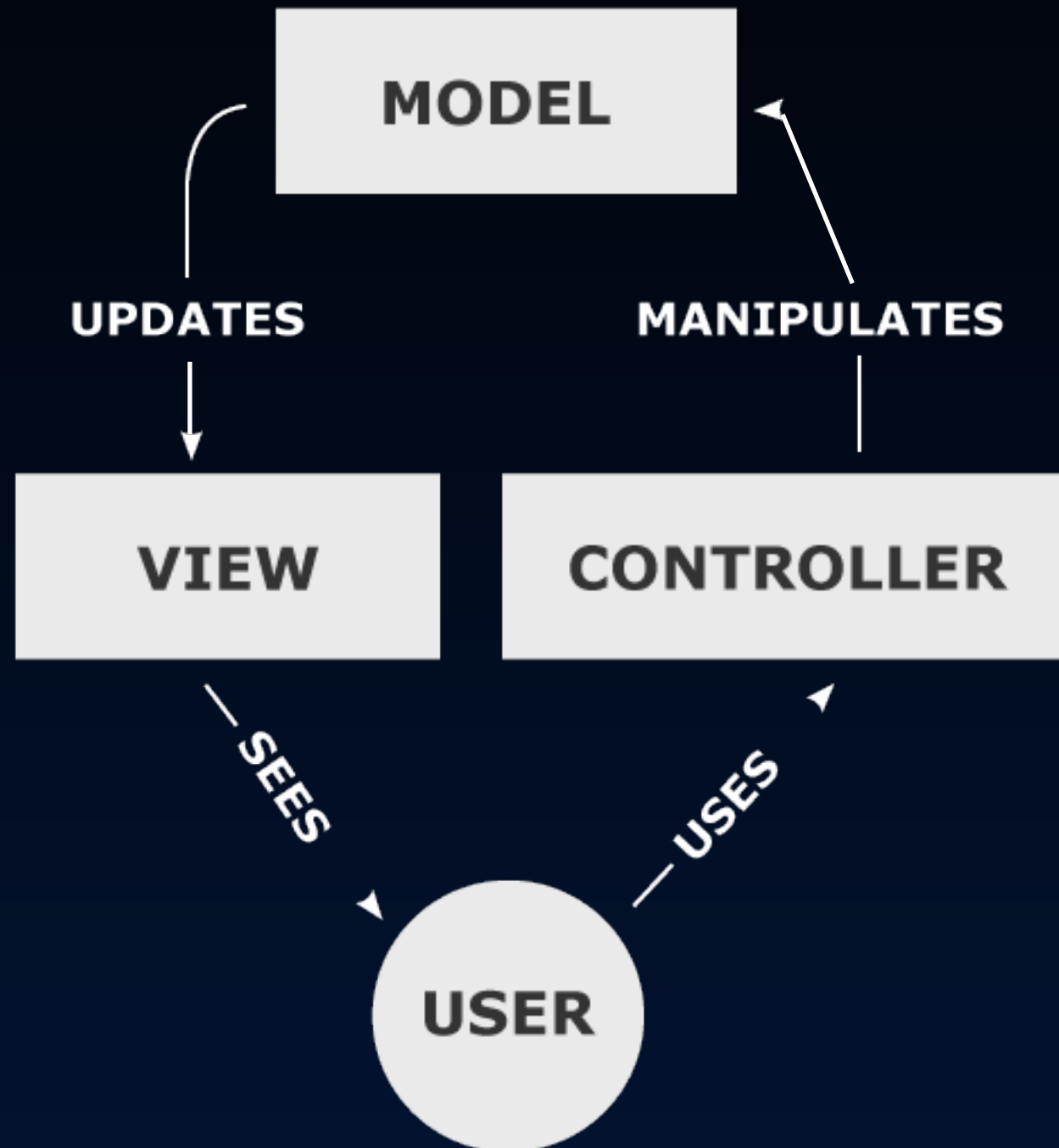
Angular.js v1 Library



<https://angularjs.org/>

Model-View-Controller

MVC pattern



Model-View-Controller

MVC pattern

- The **Model** is simply all of the data the application needs to work with.
 - In AngularJS, the Model is available to the Controller as the `$scope` object variable.
- The **View** is the presentation layer shown to the user, including controls.
 - In AngularJS, the View is the HTML document.
- The **Controller** focuses on code to work with the data in the Model.
 - In AngularJS, a Controller is simply a JavaScript function.

Angular.js v1 Library

2. AngularJS Extends HTML

AngularJS extends HTML with new attributes called *ng-directives*.

```
<div ng-app="myApp">
```

Angular.js v1 Library

3. AngularJS Applications

AngularJS *modules* define AngularJS applications.

AngularJS *controllers* control AngularJS applications.

The *ng-app* directive defines the application, the *ng-controller* directive defines the controller.

```
<div ng-app="myApp" ng-controller="myCtrl">
```

Angular.js v1 Library

4. When to Load the Library

While we often place scripts at the end of the `<body>` element, you should load the AngularJS library in the `<head>` element.

Calls to Angular can only take place after the library has been loaded.

```
<head>  
  <title>AngularJS Example</title>  
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.9/  
    angular.min.js"></script>  
</head>
```


Angular.js Concepts

- 1. Directives**
- 2. Expressions + Operators**
- 3. Modules**
- 4. Data Model**
- 5. HTML View**
- 6. Controllers**
- 7. Scope**

Directives

ng-directives are written as HTML element attributes with an **ng-** prefix.

AngularJS has a set of built-in directives which offers functionality to your applications.

AngularJS also lets you define your own directives.

```
<div ng-app="myApp">
```

Common Directives

- The **ng-app** directive defines an AngularJS application.
- The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.
- The **ng-bind** directive binds application data to the HTML view.
- The **ng-init** directive initializes AngularJS application variables. Rarely used.

*You can use **data-ng-**, instead of **ng-**, if you want to make your page's HTML code validate.*

Directives Example

```
<div ng-app="">  
  <p><input type="text" ng-model="name1"></p>  
  <p ng-bind="name1"></p>  
</div>
```

The **ng-app** directive tells AngularJS that the `<div>` element is the "owner" of an AngularJS application.

The **ng-model** directive binds the value of the input field to the application variable *name1*.

The **ng-bind** directive binds the JavaScript *innerHTML* property of the `<p>` element to the application variable *name1*.

Repeating HTML Elements with `ng-repeat` Directive

```
<div ng-app="" ng-init="names=['Jani','Hege','Kai']">  
  <ul>  
    <li ng-repeat="x in names">  
      {{ x }}  
    </li>  
  </ul>  
</div>
```

The `ng-repeat` directive repeats an HTML element and its children.

The `ng-repeat` directive actually clones HTML elements once for each item in a collection.

Expressions and Operators

AngularJS expressions are written inside double braces.

```
{{ 5 + 5 }}
```

```
{{ firstName + " " + lastName }}
```

AngularJS will output data exactly where the expression is written.

```
<div ng-app="">
```

```
  <p>My first expression: {{ 5 + 5 }}</p>
```

```
</div>
```

Expressions and Operators

AngularJS expressions are much like JavaScript expressions.

They can contain literal values, operators, and variables.

Numbers, strings, arrays and object values are core JavaScript.

```
<div ng-app="" ng-init="myCol='lightblue'">  
  <input style="background-color:{{myCol}}"  
    ng-model="myCol" value="{{myCol}}">  
</div>
```

AngularJS expressions can also be written inside a directive:

```
<p ng-bind="myCol"></p>
```

Modules

- **AngularJS modules define applications.**
- **The module is a container for the application and its controllers.**
- **Controllers always belong to a module.**
- **A module is created by using the AngularJS function `angular.module()`**

```
<div ng-app="myApp">...</div>  
<script>  
    var app = angular.module("myApp", []);  
</script>
```

The "myApp" parameter refers to an HTML element in which the application will run.

Next we would add a controller, a JavaScript function, to provide functionality, logic, and manipulation of our model in our application.

Data Model

The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.

With the **ng-model** directive you can bind the value of an input field to a variable created in AngularJS.

```
<div ng-app="myApp" ng-controller="myCtrl">
  Name: <input ng-model="name">
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function($scope) {
    $scope.name = "John Doe";
  });
</script>
```

Data Model

AngularJS applications usually have a data model.

The data model is a collection of data available for the application.

```
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.firstname = "John";  
    $scope.lastname = "Doe";  
});
```

`$scope` is the data model in the controller.

Data Model + HTML View

The HTML container where the AngularJS application is displayed, is called the View.

The View has access to the model, and there are several ways of displaying model data in the view.

You can use the **ng-bind** directive, which will bind the *innerHTML* of the element to the specified model property.

```
<p ng-bind="firstname"></p>
```

You can also use double curly brackets, or braces **{{ }}** to display content from the model:

```
<p>First name: {{firstname}}</p>
```

Or you can use the **ng-model** directive on HTML controls (input, select, textarea) to bind the Model to the View.

```
<input ng-model="firstname">
```

Two-way Data Binding

When data in the Model changes, the View reflects the change, and when data in the View changes, the Model is updated as well, immediately and automatically.

Because of the immediate synchronization of the Model and the View, the controller can be completely separated from the View, and focus on Model data.

```
<div ng-app="" ng-init="quantity=1;price=5">  
  Quantity: <input type="number" ng-  
model="quantity">  
  Costs: <input type="number" ng-model="price">  
  Total: {{ quantity * price }}  
</div>
```

Controllers

AngularJS controllers control applications and the data in the application's model.

AngularJS controllers are regular JavaScript functions.

The `ng-controller` directive defines the application controller.

Add a controller to your application, and refer to the controller with the `ng-controller` directive.

A controller can have properties and methods (variables as functions).

Controller Example

```
<div ng-app="myApp" ng-controller="myCtrl">
  First Name: <input type="text" ng-model="firstName"> <br>
  Last Name: <input type="text" ng-model="lastName"> <br>
  Full Name: {{firstName + " " + lastName}}
</div>
<script>
  var app = angular.module("myApp", []);
  app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
  });
</script>
```

- The AngularJS application is defined by **ng-app="myApp"**.
- The application is bound to, and runs inside the **<div>**.
- The **ng-controller="myCtrl"** attribute is an AngularJS directive, defining a controller.
- The **myCtrl** function is a JavaScript function attached with **.controller()**.

Scope

`$scope`

The scope, `$scope`, is the binding part between the HTML view and the JavaScript controller.

The scope is an object value with all available properties and methods to the application's model.

The scope is available for both the view and the controller.

When you make a controller in AngularJS, you pass the `$scope` object variable as a parameter.

Properties made in the controller can be referred to in the view via `$scope`.

Scope

\$scope

```
<div ng-app="myApp" ng-controller="myCtrl">
  <h1> {{carname}} </h1>
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function($scope) {
    $scope.carname = "Volvo";
  });
</script>
```

When adding properties to the `$scope` object in the controller, the HTML view can access these properties.

In the view, you do not use the prefix `$scope`, you just refer to a property name, like `{{carname}}`.

Understating the Scope

If we consider an **AngularJS** application to consist of:

1. **View**, which is the HTML.
2. **Model**, which is the data available for the current view.
3. **Controller**, which is the JavaScript function that modifies (makes / changes / removes / controls) the application's data.

Then the scope is the **Model**.

In the controller, the scope variable **\$scope** is a JavaScript object with properties and methods.

The scope is available to both the view (HTML) and the controller (JS function).

HTTP object for external data

\$http

```
<div ng-app="myApp" ng-controller="myCtrl">
  <h1> {{carname}} </h1>
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function($scope, $http) {
    $http.get('my.json').then( function(response) {
      $scope.prop = response.data;
    }
  ));
</script>
```

When loading external JSON data via the **\$http** object in the controller, we can easily attach a block of code to execute upon external file load via **then()** method.

Code Examples

“Building a Portfolio App”

*Using the Angular.js library
to create an interactive mobile app.*

BREAK

- 10 minute break
- **Please return by 7:00pm sharp**

What to Do Next

- **Homework**
 - **Assignment 12: Final Class Portfolio App**
 - » create an Angular.js v1 MVC application.
 - » store all details for your class projects in JSON.
 - » display thumbnail, title, description for projects.
 - » provide both browse and search functionality.
 - Homework due **uploaded** to Canvas by **11:59pm Fri 12/23**
 - ***canvas.santarosa.edu/courses/18079***

BREAK TO LAB

- 10 minute break
- **Please meet in lab at 8:00pm sharp**
- **Final Phaser Game Presentations**