Topic 17: AngularJS

CITS3403 Agile Web Development

Getting MEAN with Mongo, Express, Angular and Node, Chapter 8

Material from: Michael McCarthy, CMU

Semester 1, 2018

What is Angular

Angular is a MVC Javascript Framework by Google for Rich Web Application Development

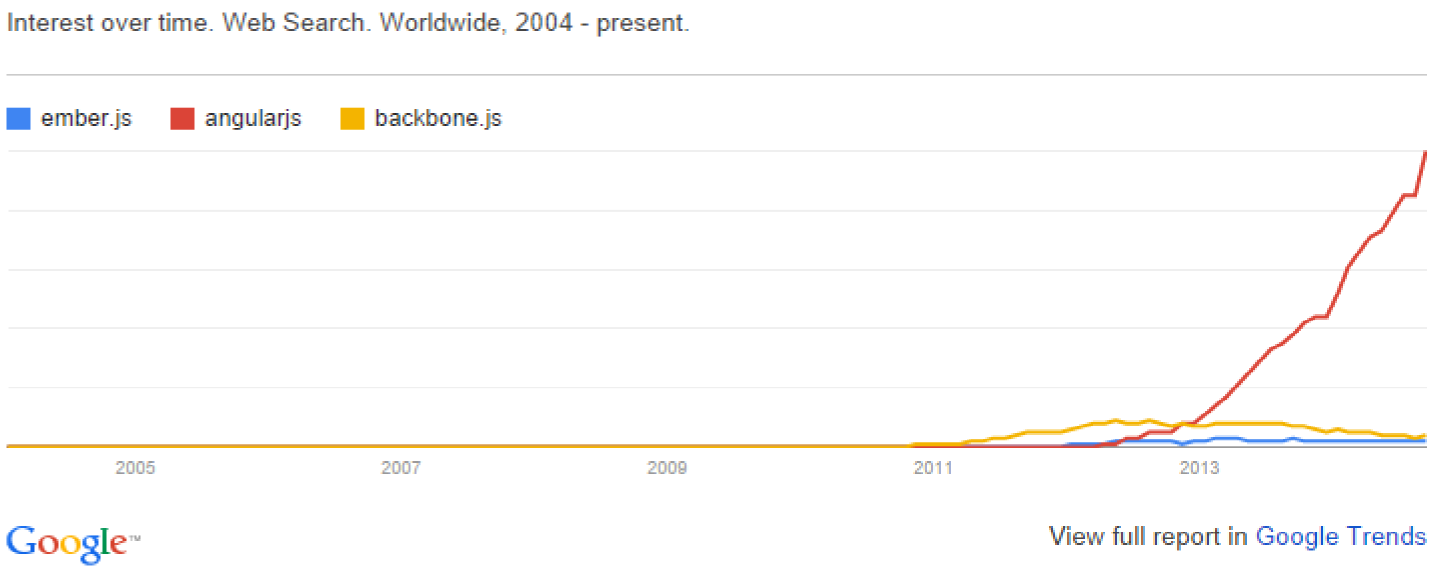
“Other frameworks deal with HTML’s shortcomings by either abstracting away HTML, CSS, and/or JavaScript or by providing an imperative way for manipulating the DOM. Neither of these address the root problem that HTML was not designed for dynamic views”.

- Structure, Quality and Organization
- Lightweight (< 36KB compressed and minified)
- Free
- Separation of concern
- Modularity
- Extensibility & Maintainability
- Reusable Components

“HTML? Build UI Declaratively! CSS? Animations! JavaScript? Use it the plain old way!”

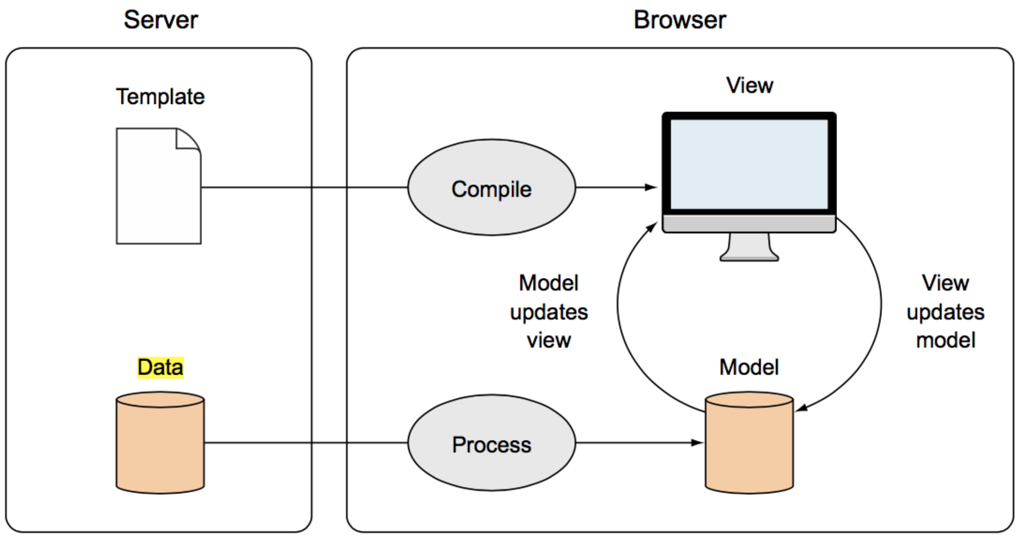
Other JS Frameworks

- BackboneJS – Models, Views, View-Models
- EmberJS – MVVM
- ReactJS – Facebook UI framework.
- JQuery –
  - Allows for DOM Manipulation
  - Does not provide structure to your code
  - Does not allow for two way binding

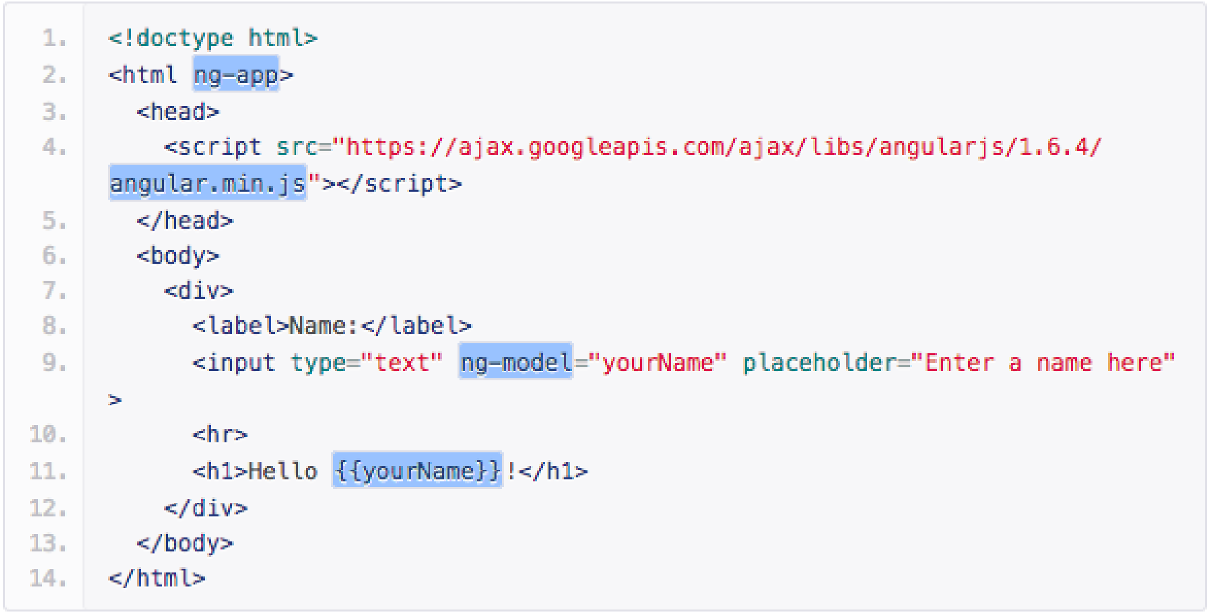


Features of AngularJS

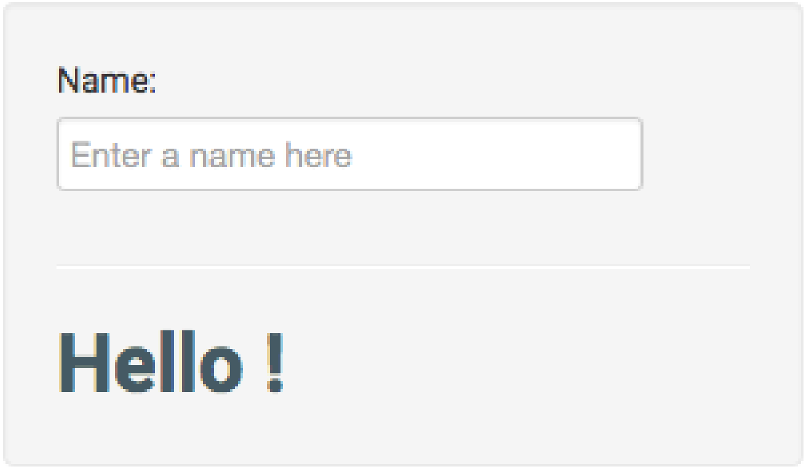
- Two-way Data Binding – Model as single source of truth
- Directives – Extend HTML
- MVC
- Dependency Injection
- Testing
- Deep Linking (Map URL to route Definition)
- Server-Side Communication



Data Binding



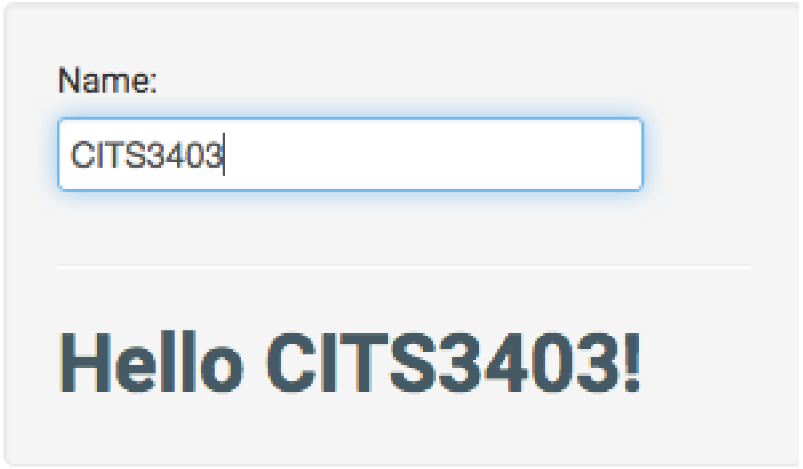
```
<!doctype html>
<html ng-app>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/
angular.min.js"></script>
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="yourName" placeholder="Enter a name here"
>
      <hr>
      <h1>Hello {{yourName}}!</h1>
    </div>
  </body>
</html>
```



Name:

Enter a name here

Hello !

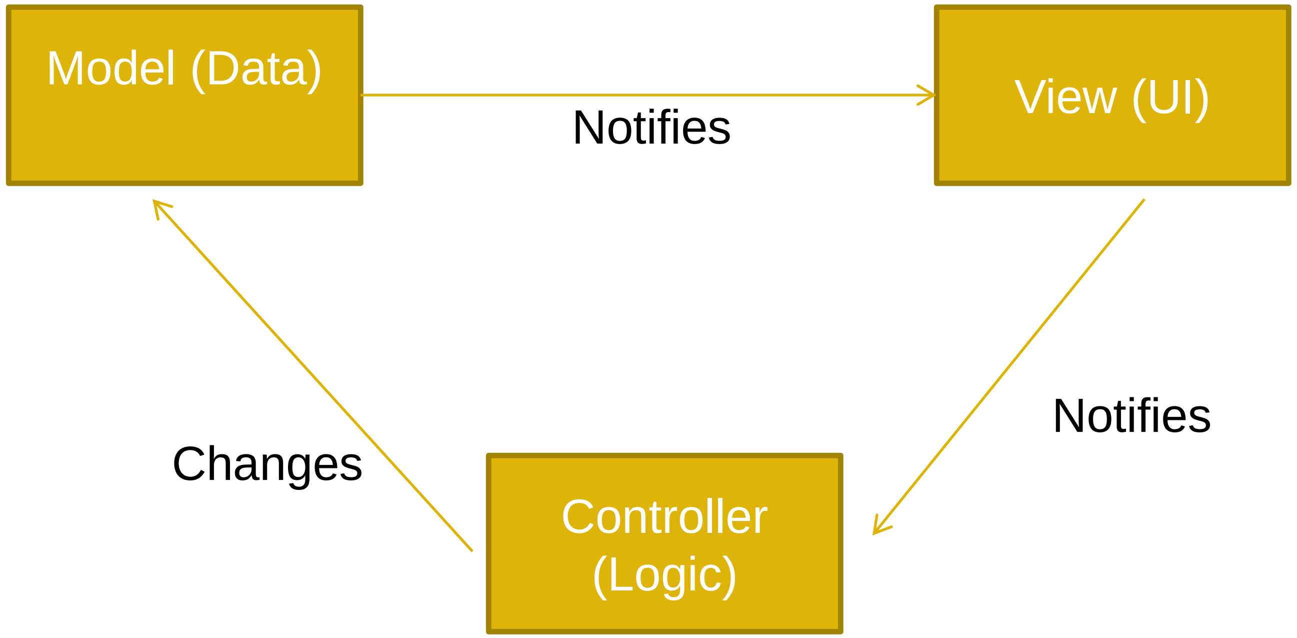


Name:

CITS3403

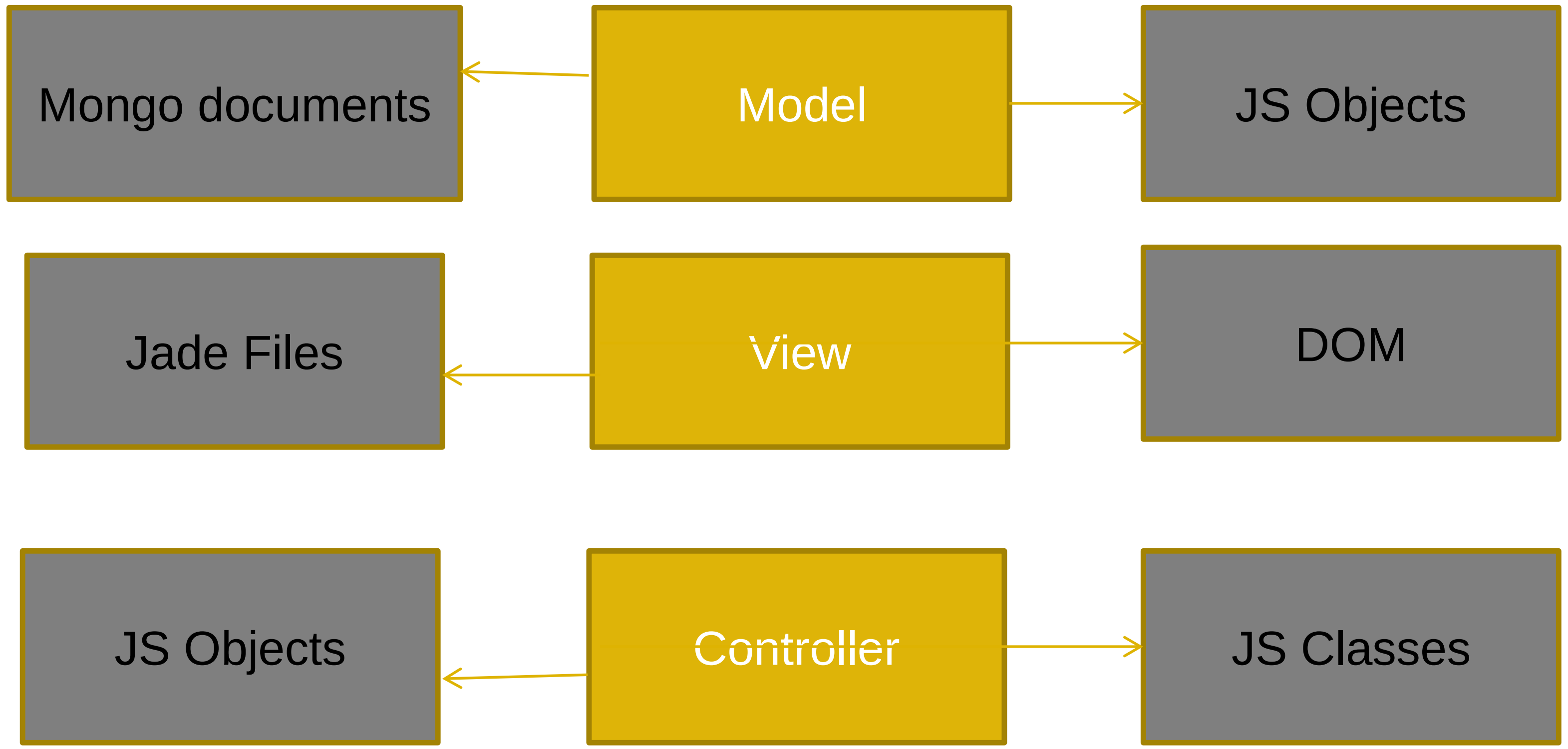
Hello CITS3403!

MVC


Model (Data)
Notifies
View (UI)
Notifies
Changes
Controller (Logic)

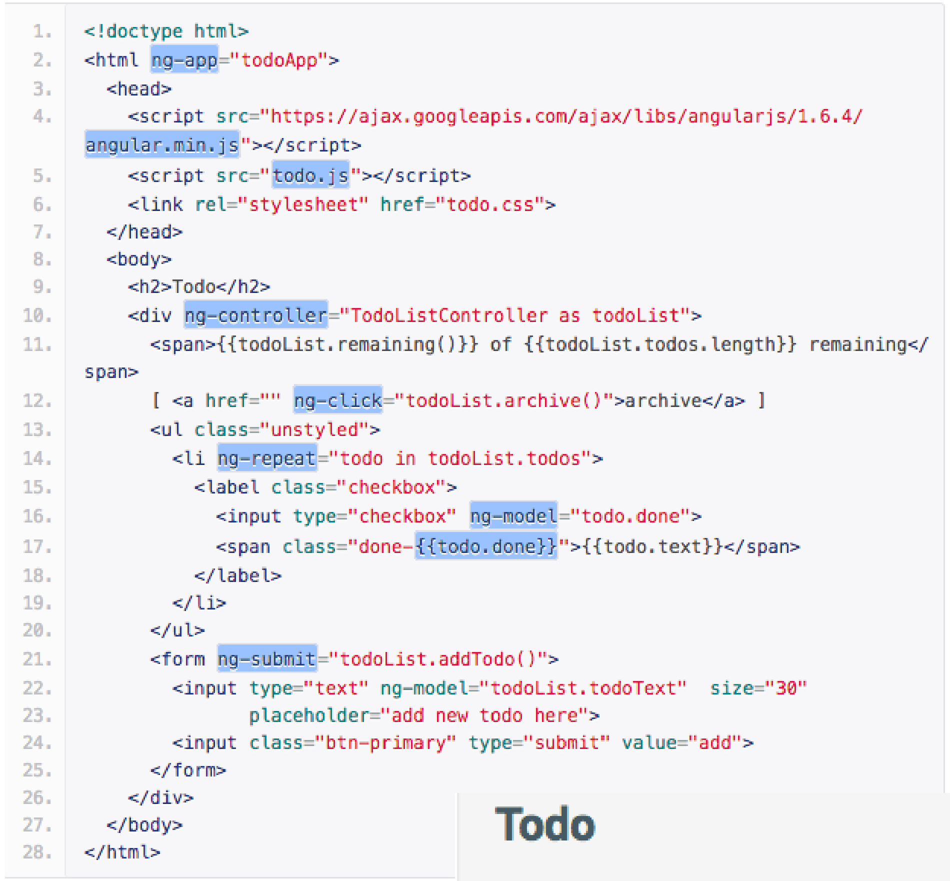
MVC

Express server side vs Angular Client Side



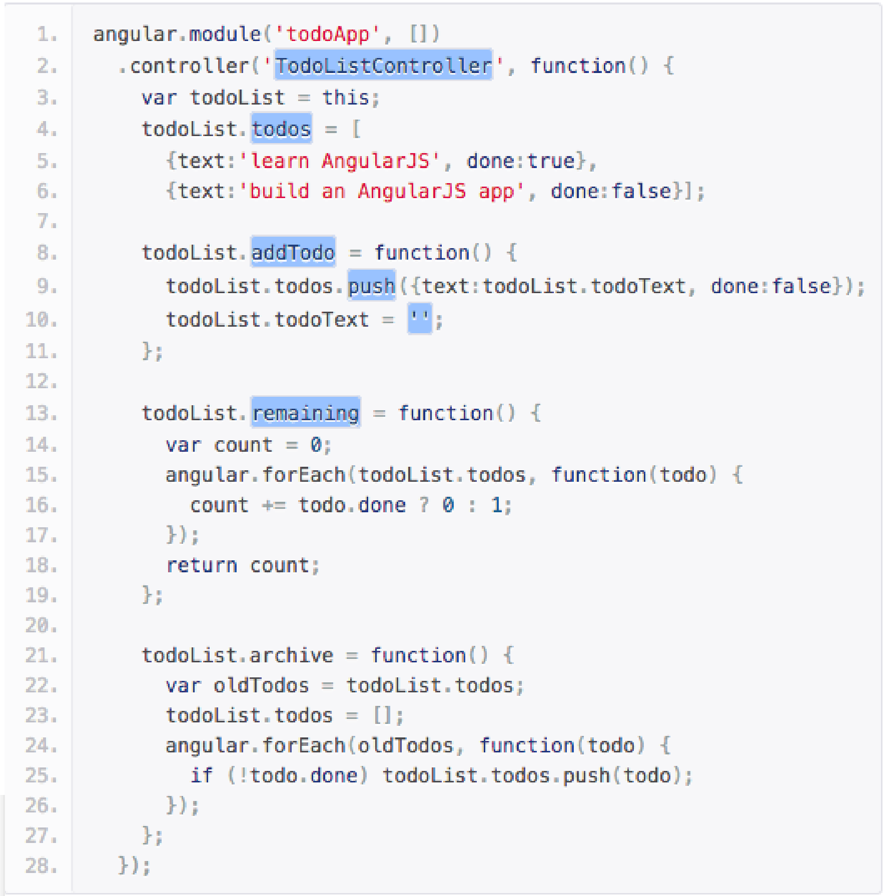
MVC

index.html



```html
<!doctype html>
<html ng-app="todoApp">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
    <script src="todo.js"></script>
    <link rel="stylesheet" href="todo.css">
  </head>
  <body>
    <h2>Todo</h2>
    <div ng-controller="TodoListController as todoList">
      <span>{{todoList.remaining()}} of {{todoList.todos.length}} remaining</span>
      [ <a href="" ng-click="todoList.archive()">archive</a> ]
      <ul class="unstyled">
        <li ng-repeat="todo in todoList.todos">
          <label class="checkbox">
            <input type="checkbox" ng-model="todo.done">
            <span class="done-{{todo.done}}">{{todo.text}}</span>
          </label>
        </li>
      </ul>
      <form ng-submit="todoList.addTodo()">
        <input type="text" ng-model="todoList.todoText"  size="30"
               placeholder="add new todo here">
        <input class="btn-primary" type="submit" value="add">
      </form>
    </div>
  </body>
</html>
```

todo.js

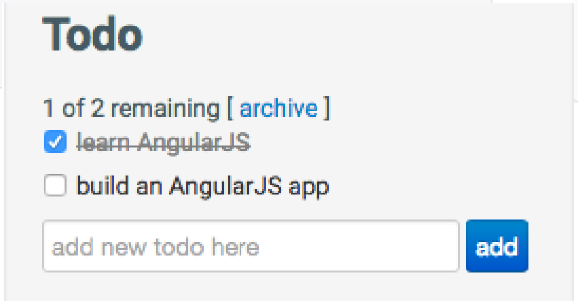


```javascript
angular.module('todoApp', [])
  .controller('TodoListController', function() {
    var todoList = this;
    todoList.todos = [
      {text:'learn AngularJS', done:true},
      {text:'build an AngularJS app', done:false}];

    todoList.addTodo = function() {
      todoList.todos.push({text:todoList.todoText, done:false});
      todoList.todoText = '';
    };

    todoList.remaining = function() {
      var count = 0;
      angular.forEach(todoList.todos, function(todo) {
        count += todo.done ? 0 : 1;
      });
      return count;
    };

    todoList.archive = function() {
      var oldTodos = todoList.todos;
      todoList.todos = [];
      angular.forEach(oldTodos, function(todo) {
        if (!todo.done) todoList.todos.push(todo);
      });
    };
  });
```



Todo

1 of 2 remaining [archive]

✓ ~~learn AngularJS~~

☐ build an AngularJS app

Hello

HTML:

```
<p>Hello World!</p>
```

JQuery:

```
<p id="greeting2"></p>
<script>
$(function(){
  $('#greeting2').text('Hello
     World!');
});
</script>
```

Angular:

```
<p ng:init="greeting = 'Hello
    World!'">{{greeting}}</p>
```

JavaScript:

```
<p id="greeting1"></p>

<script>
var isIE = document.attachEvent;
var addListener = isIE
  ? function(e, t, fn) {
      e.attachEvent('on' + t, fn);}
  : function(e, t, fn) {
      e.addEventListener(t, fn, false);};
addListener(document, 'load', function(){
  var greeting =
     document.getElementById('greeting1');
  if (isIE) {
    greeting.innerText = 'Hello World!';
  } else {
    greeting.textContent = 'Hello World!';
 }
});
</script>
```

Angular Concepts

Template	HTML with additional markup used to describe what should be displayed
Directive	Allows developer to extend HTML with own elements and attributes (reusable pieces)
Scope	Context where the model data is stored so that templates and controllers can access
Compiler	Processes the template to generate HTML for the browser
Data Binding	Syncing of the data between the Scope and the HTML (two ways)
Dependency Injection	Fetching and setting up all the functionality needed by a component
Module	A container for all the parts of an application
Service	A way of packaging functionality to make it available to any view

Templates, Scopes and Controllers

- Best practice: Each **template** component gets a new **scope** and is paired with a **controller**.
- **Expressions** in templates:
  - {{foo + 2 * func()}} are evaluated in the context of the scope.
  - Controller sets up scope: $scope.foo = ... ; $scope.func = function() { ... };
- Best practice: Keep expressions simple put complexity in controller
- Controllers make model data available to view template
- A scope object gets its prototype set to its enclosing parent scope
  -
    ```
    <div ng-controller="crtl1">
       <div ng-controller="crtl2"> ...
       </div>
    </div>
    ```
- ScopeB's prototype points at ScopeA. Useful since scopes are frequently created (e.g. ng-repeat, etc.)

Scope watches

- Two-way binding works by watching when expressions in view template change and updating the corresponding part of the DOM.

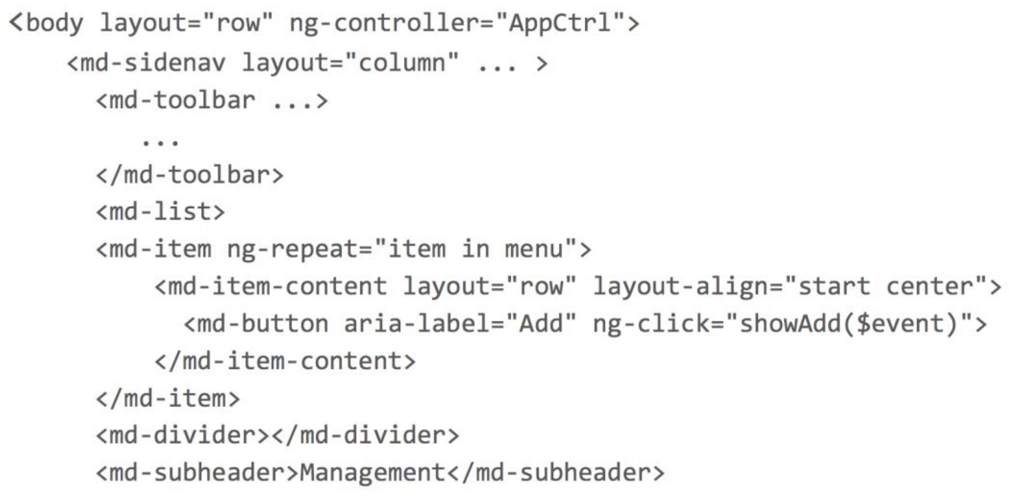
- Angular add a **watch** for every variable or function in template expressions

- During the **digest** processing all watched expressions are compared to their
  - previously known value and if different the template is reprocessed and the
  - DOM update
  - ○ Angular automatically runs digest after controller run, etc.
  - It is possible to:
    Add your own watches: ($scope.$watch(..)) (e.g. caching in controller)
  - Trigger a digest cycle: ($scope.$digest()) (e.g. model updates in event)

Directives

- Angular preferred method for building reusable components
  - Package together HTML template and Controller and extend templating language.
  - Ng prefixed items in templates are directives

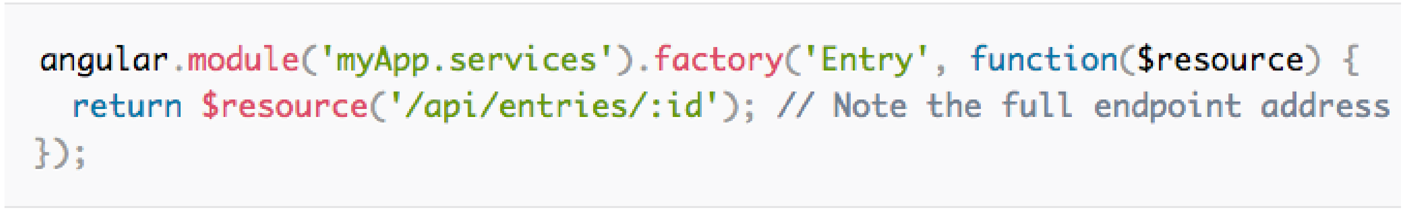
- Directive can:
  - Be inserted by HTML compiler as:
    - attribute (<div my-dir="foo">...</div>)
    - element (<my-dir arg1="foo">...</my-dir>)
  - Specify the template and controller to use
  - Accept arguments from the template
  - Run as a child scope or isolated scope



```
<body layout="row" ng-controller="AppCtrl">
    <md-sidenav layout="column" ... >
      <md-toolbar ...>
          ...
      </md-toolbar>
      <md-list>
      <md-item ng-repeat="item in menu">
          <md-item-content layout="row" layout-align="start center">
            <md-button aria-label="Add" ng-click="showAdd($event)">
          </md-item-content>
      </md-item>
      <md-divider></md-divider>
      <md-subheader>Management</md-subheader>
```

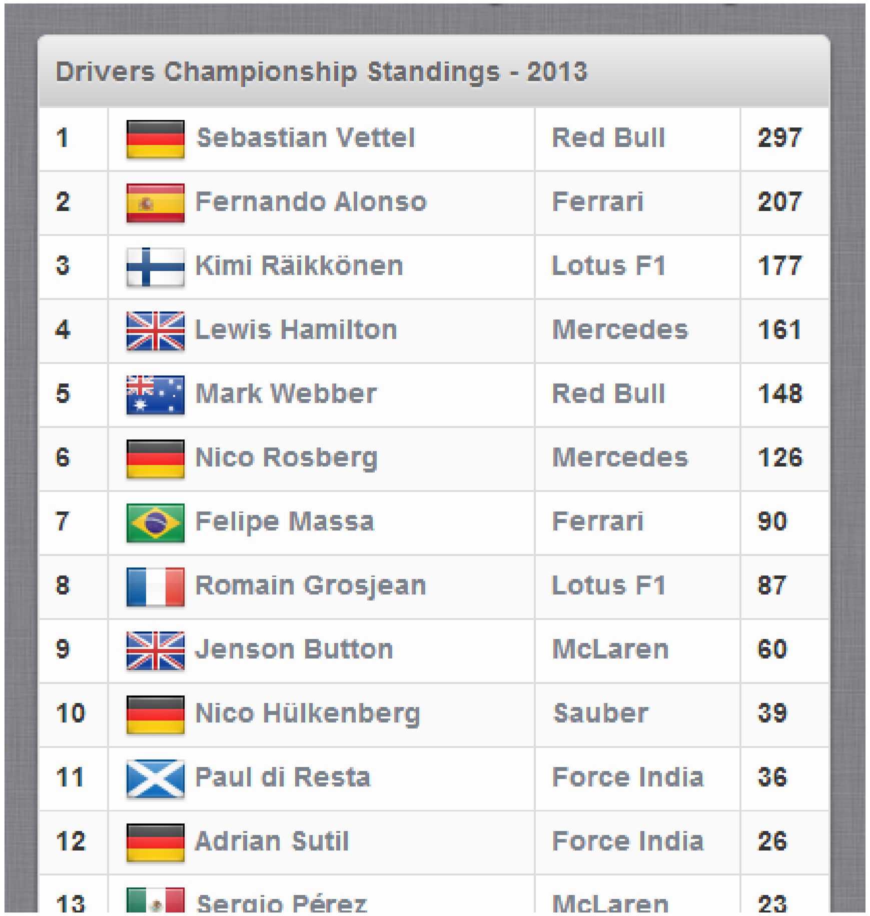
Services

- Used to provide code modules across view components
  - Example: shared JavaScript libraries
- Angular has many built-in services
  - Server communication (model fetching)
    - $http, $resource, $xhrFactory
  - Wrapping DOM access (used for testing mocks)
    - $location, $window, $document, $timeout, $interval
  - Useful JavaScript functionality
    - $animate, $sce, $log
  - Angular internal accesses
    - $rootScope, $parse, $compile



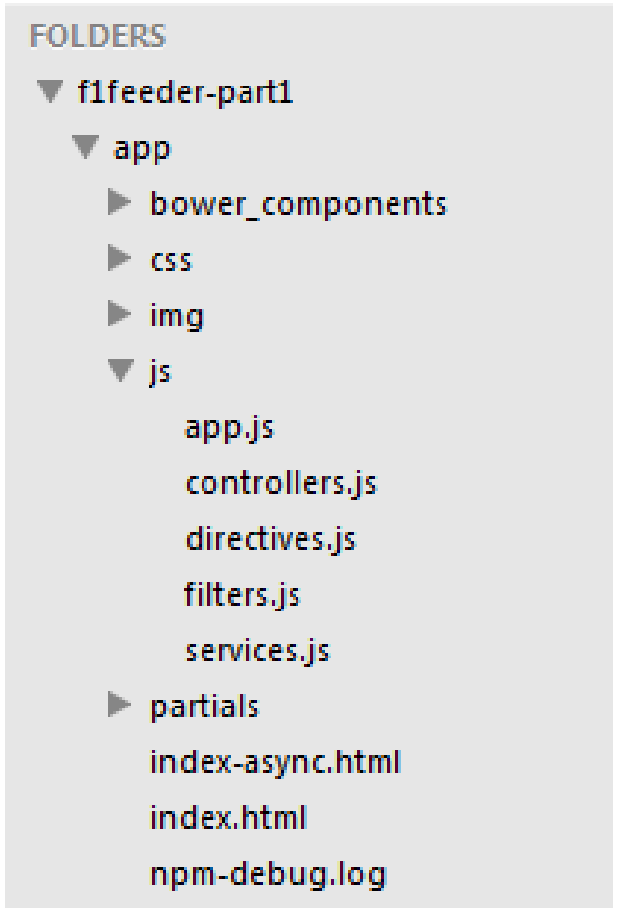
```
angular.module('myApp.services').factory('Entry', function($resource) {
  return $resource('/api/entries/:id'); // Note the full endpoint address
});
```

Example App



Drivers Championship Standings - 2013

1	Sebastian Vettel	Red Bull	297
2	Fernando Alonso	Ferrari	207
3	Kimi Räikkönen	Lotus F1	177
4	Lewis Hamilton	Mercedes	161
5	Mark Webber	Red Bull	148
6	Nico Rosberg	Mercedes	126
7	Felipe Massa	Ferrari	90
8	Romain Grosjean	Lotus F1	87
9	Jenson Button	McLaren	60
10	Nico Hülkenberg	Sauber	39
11	Paul di Resta	Force India	36
12	Adrian Sutil	Force India	26
13	Sergio Pérez	McLaren	23



```
FOLDERS
▼ f1feeder-part1
  ▼ app
    ▶ bower_components
    ▶ css
    ▶ img
    ▼ js
        app.js
        controllers.js
        directives.js
        filters.js
        services.js
    ▶ partials
      index-async.html
      index.html
      npm-debug.log
```

From https://www.toptal.com/angular-js/a-step-by-step-guide-to-your-first-angularjs-app

Sample Angular Powered View

```
<body ng-app="F1FeederApp" ng-controller="driversController">
  <table>
    <thead>
      <tr><th colspan="4">Drivers Championship Standings</th></tr>
    </thead>
    <tbody>
      <tr ng-repeat="driver in driversList">
        <td>{{$index + 1}}</td>
        <td>
          <img src="img/flags/{{driver.Driver.nationality}}.png" />
          {{driver.Driver.givenName}}&nbsp;{{driver.Driver.familyName}}
        </td>
        <td>{{driver.Constructors[0].name}}</td>
        <td>{{driver.points}}</td>
      </tr>
    </tbody>
  </table>
</body>
```

Expressions

Expressions allow you to execute some computation in order to return a desired value.

- {{ 1 + 1 }}
- {{ 946757880 | date }}
- {{ user.name }}

you shouldn’t use expressions to implement any higher-level logic.

Directives

Directives are markers (such as attributes, tags, and class names) that tell AngularJS to attach a given behaviour to a DOM element (or transform it, replace it, etc.)

Some angular directives

- The ng-app - Bootstrapping your app and defining its scope.
- The ng-controller - defines which controller will be in charge of your view.
- The ng-repeat - Allows for looping through collections

Adding Controllers

```
angular.module('F1FeederApp.controllers', []).
controller('driversController', function($scope) {
    $scope.driversList = [
      {
         Driver: {
             givenName: 'Sebastian',
             familyName: 'Vettel'
          },
          points: 322,
          nationality: "German",
          Constructors: [
             {name: "Red Bull"}
          ]
      },
      {
          Driver: {
          givenName: 'Fernando',
             familyName: 'Alonso'
          },
          points: 207,
          nationality: "Spanish",
          Constructors: [
             {name: "Ferrari"}
          ]
      }
    ];
});
```

- The $scope variable – Link your controllers and view

App.js

```
angular.module('F1FeederApp', [
  'F1FeederApp.controllers'
]);
```

Initializes our app and register the modules on which it depends

Index.html

```
<body ng-app="F1FeederApp" ng-controller="driversController">
  <table>
    <thead>
      <tr><th colspan="4">Drivers Championship
       Standings</th></tr>
    </thead>
    <tbody>
      <tr ng-repeat="driver in driversList">
        <td>{{$index + 1}}</td>
       <td>
      <img src="img/flags/{{driver.Driver.nationality}}.png" />
       {{driver.Driver.givenName}}&nbsp;{{driver.Driver.familyName}}
        </td>
        <td>{{driver.Constructors[0].name}}</td>
        <td>{{driver.points}}</td>
      </tr>
    </tbody>
  </table>
<script
       src="bower_components/angular/
       angular.js"></script>
  <script
       src="bower_components/angular-
       route/angular-route.js"></script>
  <script src="js/app.js"></script>
  <script src="js/services.js"></script>
  <script
       src="js/controllers.js"></script>
</body>
</html>
```

Loading data from the server

```
angular.module('F1FeederApp.services', []).
  factory('ergastAPIservice', function($http) {

    var ergastAPI = {};

    ergastAPI.getDrivers = function() {
      return $http({
        method: 'JSONP',
        url: 'http://ergast.com/api/f1/2013/driverStandings.json?callback=JSON_CALLBACK'
      });
    }

    return ergastAPI;
  });
```

- $http - a layer on top of XMLHttpRequest or JSONP
- $resource - provides a higher level of abstraction
- Dependency Injection

we create a new module (F1FeederApp.services) and register a service within that module (ergastAPIservice).

Modified controller.js

```
angular.module('F1FeederApp.controllers', []).
  controller('driversController', function($scope, ergastAPIservice) {
    $scope.nameFilter = null;
    $scope.driversList = [];

    ergastAPIservice.getDrivers().success(function (response) {
        //Dig into the response to get the relevant data
        $scope.driversList = 
      response.MRData.StandingsTable.StandingsLists[0].DriverStandings;
    });
  });
```

Routes

- $routeProvider – used for dealing with routes

Modified app.js

```
angular.module('F1FeederApp', [
  'F1FeederApp.services',
  'F1FeederApp.controllers',
  'ngRoute'
]).
config(['$routeProvider', function($routeProvider) {
  $routeProvider.
      when("/drivers", {templateUrl: "partials/drivers.html", controller:
      "driversController"}).
      when("/drivers/:id", {templateUrl: "partials/driver.html", controller:
      "driverController"}).
      otherwise({redirectTo: '/drivers'});
}]);
```

Partial views

```
<!DOCTYPE html>
<html>
<head>
  <title>F-1 Feeder</title>
</head>

<body ng-app="F1FeederApp">
  <ng-view></ng-view>
  <script src="bower_components/angular/angular.js"></script>
  <script src="bower_components/angular-route/angular-route.js"></script>
  <script src="js/app.js"></script>
  <script src="js/services.js"></script>
  <script src="js/controllers.js"></script>
</body>
</html>
```

Bower is a package manager for client side artifacts…