**AngularJS Overview**

* Views are rendered in AngularJs as a combination of templates and controllers
* Templates are built using directives
* Data filtering using **filters**
* Controllers pull the data in using **factories, services, and providers**
* **Factories, services, and providers** interact with the server to pull in the data into application
* Angular application is organized as a module or as a group of cooperating modules. Each module has a configuration. **Routes** take care moving from one view to other.

**Front-End JS Frameworks**

* Well defined architectures (MVC, MVVM) helps with complexity of managing DOM manipulation and data updates manually
* Software Library:
  + Collection of implementation of behavior with a well defined interface by which the behavior is revoked
  + E.g. jQuery
* Software Framework
  + Abstraction in which software provides a generic functionality and you will writing you our code to add further to that functionality
  + Provides reusable environment that provides particular functionality
  + E.g. Angular, Ember, Backbone
* Library vs Framework
  + **Library:** a collection of functions, which are useful when writing web apps. You code is in charge and it calls into the library when sees fit.
  + **Frameworks:** a particular implementation of a web application, where your code fills in the details. The framework is in charge and it calls into your code when it needs something app specific.
    - Hollywood principle: Don’t call us, we’ll call you! (Framework is in control)
    - Inversion of Control
    - Imperative vs. Declarative Programming
      * Traditionally we use **imperative** programming approach. We specify steps to be done and specify them in sequence. You’re in control to how the steps will be executed.
      * In **declarative** programming, you’re specifying what needs to be done but don’t care how. You leave that on framework to execute what needs to be done.
* Terms we’ll be hearing a lot in this course
  + Single Page Application/Rich Internet Aplication
  + Model-View-Controller
    - Data binding, routing
  + Scalable, Reusable, Maintainable JS code
  + Test driven development
* Popular JS Frameworks
  + Angular, Ember, Backbone, React, Aurelia, Meteor, Polymer (from Google), Knockout, Vue, Mercury
* Ember vs. Angular vs. Backbone
  + ***Backbone*** provides most flexibility and ***Ember*** is most opinionated. As for Ease of use ***Ember*** is hardest and ***Backbone*** is the easiest. Leaning curve ***Backbone*** is easy to start with and ***Ember*** is hardest. ***Angular*** sits in the middle for **Flexibility**, **ease of use**, and **learning curve**, hence it’s the most popularly used out of the three.

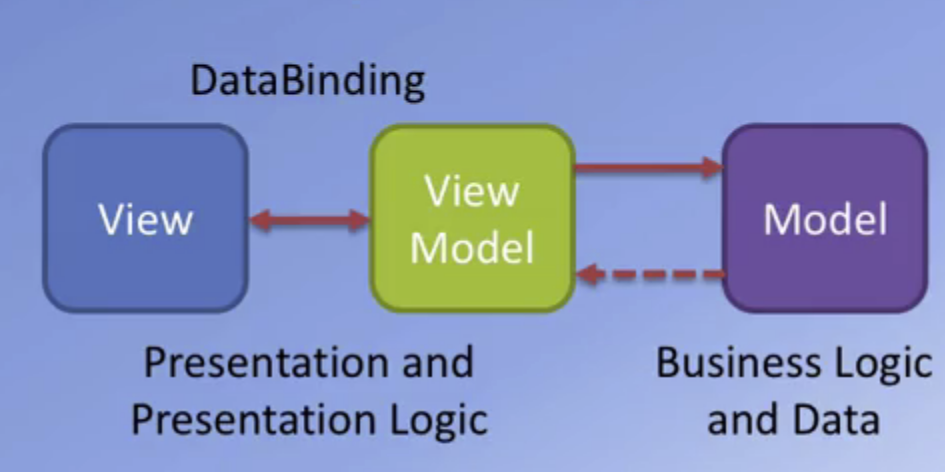
***Introduction to AngularJS***

* First released in 2012 by a Google employee Misko Hevery
* Structural framework for dynamic web applications
  + HTML only display static documents, but Angular fills in the gap to make the HTML dynamic
  + Angular works well with CRUD (date-driven) applications
    - CRUD: Create, read, update, and delete
* Important Angular Vocabulary
  + Two-way Data Binding
  + Scope
  + Directives
  + Templates
  + Routing
  + Testing
  + Modules
  + Controllers
  + Filters
  + Factory
  + Service
  + Provider
* Angular Built-in Directive
  + HTML custom attributes
    - Data-\* attributes (Bootstrap/jQuery)
    - ng-\* attributes/ data-ng-\* (for **Angular**)
  + Examples:
    - ng-app, ng-bind, ng-model, ng-init, ng-repeat, etc
    - ngApp, ngBind, ngModel, ngInit, ngRepeat, etc
      * Camel case converted to “–” (ngApp same as ng-app)
  + Directives are **declarative** programming in action
    - ng-app – Start up the Angular app
    - ng-init – Ask angular to initialize some variable or execute an expression
* The ngApp directive
  + It is applied to a HTML tag to specify the root of the application.
  + Applying ng-app to <html> tag means the entire page is under the control of the Angular application
* The ngInit directive
  + Evaluate an expression
  + Initialize a JS variable
    - <p ng-init= “index=1”></p>
    - <div class=”row” ng-init=“dish= {name:‘Test’, …}”></div>
* The ngModel Direction
  + Binds the input value to a variable within the scope (Very useful in input boxes)
    - Two-way data binding
* The ngRepeat Directive
  + This directive is a looping construct
  + Loops over items in a collection
  + Instantiates a template for each item
  + Example:
    - <ul>

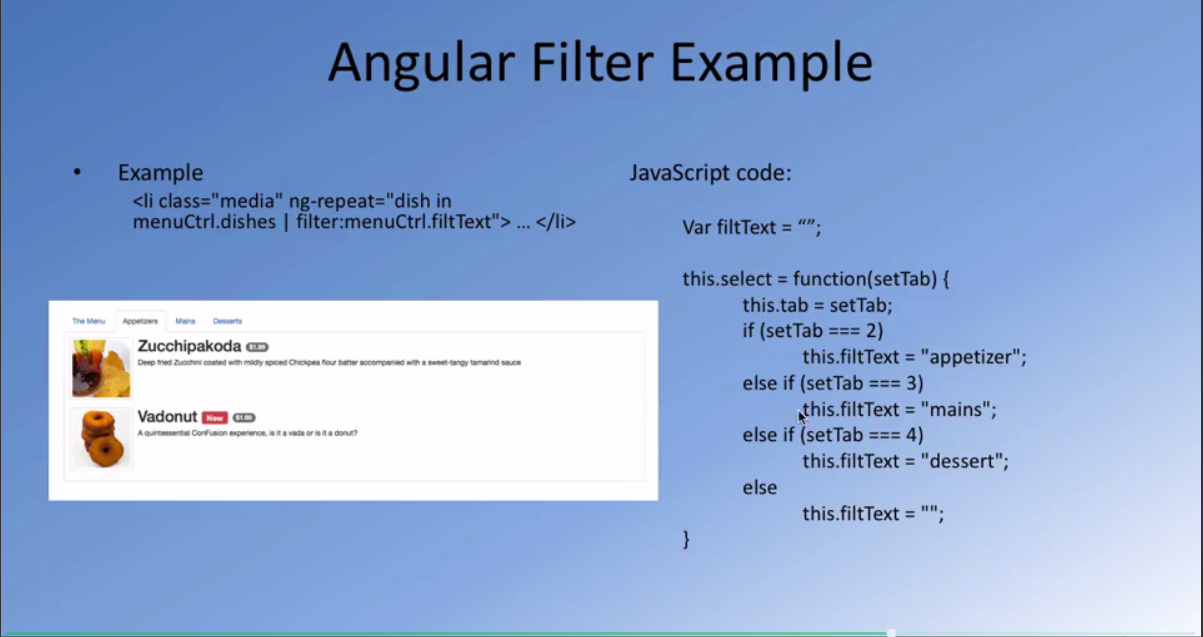
<li ng-repeat=“dish in dishes”>

* Two-way data binding
  + Bind a HTML or CSS property to a JS variable
  + When the value of the variable is updated the HTML/CSS property is also updated and vise versa
* Angular Expressions
  + Simple JS Expressions
    - Evaluated against against an Angular **scope** object
    - Expressions enclosed in {{ }}
  + Example: Expression with **scope** object
    - {{dish.name}}, {{dish.description}}

***Models, Views, and Controllers (MVC)***

* Software design pattern
  + Reusable solution to commonly occurring problems
* MVC is a software engineering architecture pattern
  + It isolates the domain logic from UI
  + Separation of Concerns
    - Independent, testing, and maintenance of the different parts of the app
* View: Presenting the info to user
  + Renders the model into a form suitable for interaction
  + Multiple views can exist for a single model for different purposes
  + A viewport typically has a one to one correspondence with the display surface
* Model: Stores the domain state/logic of the app
  + Manages the behavior and data of the app
  + Responds of the requests for info about it’s state (typically when view wants to update itself)
  + Responds to instructions to the change state (usually issued by the controller
  + Notifies all the observers (Views) for the model. When the model is updated, view is automatically notified about the changes in the model.
* Controller: Mediates between view and model
  + Receives information from users (through user input etc) and initiates a change in the state of the model
  + Accepts input from the user and instructs the model. Simultaneously, instruct the view to change the display of the info on the view.
* Model View-View Model (MVVM)
  + It is the descendent of MVC
  + 
* Angular Modules
  + Structure you code for readability of a big Angular project. **Angular module** comes in handy for such cases
  + Angular module is a collection of:
    - Controllers
    - Directives
    - Filters
    - Services
    - Other Configuration Information
  + It help keep code organized, maintainable, and easily testable
  + Angular Module Example
    - <html app=“angularAPP”>
    - In JS
      * var app=angular.module(‘angularAPP’, []);
* Angular Controller
  + JS object containing attributes/properties and functions to be used in angular expressions and directives
  + Controller example
    - <div ng-controller=“menuController as menuCtrl”>
    - In JS
      * app.controller(‘menuController’, function() {});
      * app is coming from defined in angular module

***Filters***

* It allow us to format the value of an expression before it is displayed to the end user. It can be used in view ***templates***, ***controllers***, or ***services***
* Use of filter in template
  + In an expression
    - {{ dish.price | currency }}
    - | tells the user to apply currency filter
  + With js code
    - 
* Built in Angular Filters
  + uppercase/lowercase
  + currency
  + date
  + filter
    - selects a subset of an array based on criteria specified and returns a new array
  + orderBy
  + json
  + limitTo