Impart knowledge on whatever I know.

What: HTML 6; Model View Whatever

Kind of Futuristic framework for web application development.

Belongs to Class of FWs called- data binding frameworks: Main Purpose-> SPAs uses AJAX to fetch data without reloading the part

So part of url after # changes.. so true with back button as well.

Mostly Data binding fw are based on MVC concept. And these all are mostly build SPAs, So is Angular.

So there are 3 main thing that all Data Binding fw provide:

* Routing: updating the page in response to changes in the URL, but just the portion of the URL after #. -🡪 Part of the URL changes after #/ doesn’t cause page to reload, even with AJAX calls. Changes the view.
* Templating: Dynamically Creating and updating HTML based on templates and models

(Parts of the view gets replaced later based on the model)

* Data Binding: synchronize the model and UI(view)

Other Data Binding FW: Angular Knockout, Ember.js, Can.js

Hard to tell which one’s the best. Lot of ppl prefer angular. All

**Backbone**: Library, One if the first MVC lib, lot of manual work, lot of getters and setters.

Lots of ppl said these are the things that should have been present in JS itself. Classes and inheritance. Routing Sync with backend. Lot of boiler plates involved.

Based on other libraries

Underscore – functional programming primitive

Angular based on (influential)

* JQuery(JQLite – DOM Manipulation), - JQurey like API.. access to DOM - Minimum
* AMD(split JS into many files and declare dependency -- RequireJS help in loading in proper order -- Async Module Dependency –inspire by Dependency Injection of angular), and evaluating when their dependency has been evaluated. In Ang, we see this thing depends this other thing
* Promises(Q liber by KrisKowal – The Most prominent library for promises – inspired by $q service), - fetch data from backened, Code becomes async.. start using backend. More async-> more complex. Q -> This is The Most Promising Library for promises. kinda good to be familier – as you’ll see in mostly all type of context
* Handlebars(Templating Engine – Moustache (one of the first templating libraries, n improvement of handle bars which angular has used as well))

Why Angular?

* MVC Design Pattern
  + M- Data (Javascript Object) behind the app, just the data, no presentation logic or layout
  + V- Graphical Presentation of the model
  + C- User interacts with the things on the screen and all these interaction change the model through whats called a controller. C is set of fns that change the model and may have some logic.
  + AngularJS’s MVC – MVVM (Model-View-ViewModel)
  + ViewModel – Specific Data to Specific View ($scope – obj model)
    - This scope shows only data whichever data is valid for that particular view. Nothing else. So. Model is specific to the view, and vice-versa.
* SPA Support ( build only SPAs)
  + Routing
  + No refresh of the page
* Testing
  + Unit Testing (modularized)
  + 1600 AngularJS tests in about 3 seconds
  + Jasmine
* MobileFirst
  + even if you are not building a mobile app, RESPONSIVE
* Javascript
  + Clientside Rendering
  + Perceived Performance
  + POJO (**Plain Old Java Object,** is a normal Java Object class(that is not a JavaBean, EntityBean etc.) and does not serve any special role nor does it implement any special interfaces of any of the Java frameworks )
  + Atwood’s Law: “Any Application that can be written in JS will eventually be written in JS” by Jeff Atwood, cofounder of Stack Overflow & Stack Exchange
* Google
  + Google Product: Steller level & quality of support
  + Google Backing: Increased credibility & interest
* Core AngularJS Features:
  + 2 Way Binding
  + Filters & Dynamic Templating
  + Directives

DEMO1:

1. Simple HTML Page input and Hello World
2. Console.log(Got KeyUp Event) with add event Listener for keyup event – Vanilla JS DOM API Way, no /\_ng
3. Print input value to console (textInputElement.value())
4. Assign value to inner HTML  
   So that was Straight JavaScript, No libraries, Nothing  
     
   Now same ex with JQury,jus to use how jq improved the basic vanilla js DOM API
5. .on and .html (import jq)
6. --------------------- Backbone
7. Angular
   1. Model is there by default here in angular
   2. Angular directives: expands vocab of html, teaches angular new tricks, -> most powerful probably
   3. Ng-app : bootstraps itself and say hey this is angular application. Multiple ng apps in
   4. Ng-model in input
   5. Data binding name – automatically updates the template
   6. Bound to rootScope for input
   7. No need to declare property before hand, just start using it. Ng tried to minimize the Boiler plate
8. Multiple Input bound to Same ng-model (Pain in jQ cz would have to do with all possible pairs but better in Backbone) //2Way Binding
9. {{firstname}} {{lastname}}
10. Angular Controller //Angular 1.3 + no longer we can use this
    1. function NameCtrl($scope){ (Ctlrl angular convention – , here $scope object is the model here.. its just a plain JS Object)   
        $scope.firstName = ‘John’;  
        $scope.lastName = ‘Smith’;  
       }
11. Above pollutes variables with global namespace. Introduce angular.module **(DI Later)**
12. -------------------- How does angular know when you change the property once they are set.!!  
    $scope.$watch()
13. --------------------$scope.$apply()
14. Ng-repeat ($scope.names = [“heyy”,”heyy1”,”heyy2”] in li) inside controller {for-each}  
    ------------------------------------------------------------------------------------------------------------------------------  
    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Day 2:**

1. Add element to a list ($scope.names.push) using ng-submit [normally refreshes but using angular no refresh ] – also pushes no page refres, ng-submit--- fn call, auto- $apply— auto detect updates
2. Remove Text from input field ? how do you think should I do that? Change the model
3. Remove element from the array using ng-click  
   $scope.removeName = function(name){  
    var I = $scope.names.indexOf(name);  
   $scope.names.splice(i,1);  
   }  
   <a href=”” ng-click=”removeName(name)”>remove</a> [a to make look it look as a link to be clicked on]
4. Now lets see how to use Objects instead of strings in an array. (Enumerating Objects – countries and their population) --🡪 country.name [dot as it is an angular expression]
5. Table of the same [angular + with tables]
6. Now $http – DI---🡪 Minification-🡪 get angular api
7. Production Ready… 2nd argument is an array
8. Angular Search using Angular Filters ---🡪 Explain FILTERS ( | filet:query)
9. Sorting 🡪 orderby ( |orderBy: “-population” )
10. Sorting 🡪 Decreasing order | add minus sign | order matters with pipe, yess! Just like UNIX pipes
11. Sort Interactively --🡪 <https://jsonblob.com/api/jsonBlob/5558986de4b0d557cf925fb6>
    1. Explain ORDER BY (orderBy: sortField: reverse)
12. Reverse as well
13. Add Flag URL!! ---- without ng-src [ India | China | United States of America]
14. ng-src include (load only after the template/expression evaluates) – can be on anything—like iframe
15. added Capital
16. added GDP
17. Filter | currency ()
18. Filter | number  
      
    -------------------------------------------------------------------------------------------------------------------------------\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DAY 3**

1. Simplify the page for Routing simple countries display
2. Routing using $routeProvider with inbuilt templates roting with state params
3. Routing using $routeProvider with templateURL

//Angular Expressions all inside ng-repeat and all everything

Routing -> ‘ngRoute’ --$routeProvider, config-> runs this function once in the beginning, $routeParams,

This syntax/template comes from express – a node project

$routeProvider

.When(“/”,{

Template: “<li>India</li>”

Controller: “TestCtrl”

})

.when(“/:country Name”, {

TemplateUrl: “countryList.html”

Controller: “TestingDetailCtrl”

})

.otherwise({

redirectTo: “/”

})

Factory - singleton

App.factory(“countries”, function($http){

/\*Var cachedData;

Function getData(callback){

If(cachedData){

Callback(data)

}

Else{

Get JSON and callback

}

}\*/

Return {

list: function(callback){

$http.get(“countries.json).success(callback);

}

Find: function(name, callback){

$http.get(“countries.json).success(function(data){

Var country = data.filter(function(){

Return country.name === name;

})

Callback(country);

}); 🡪 check 21.45 (2)

}

}

})

Countries.find($routeParams.name, function(country){

$scope.country = country;

})

Show DI

**Custom Filter:**

App.filter(“encodeURI”, function(){

Return window.encodeURI;

})

{{country | encodeURI}}

**Custom directive: same as prev, but encapsulated in directive**

App.directive(“myDirective”, function(){

Return{

Scope: {

Country: “=”

},

Restrict: “AECM”,

templateUrl: “abcd.hrml”,

(CHECK 45:57) -> controller: function($scope, countries){

Console.log($scope.country);

}

}

})

<li ng-repeat=”country in counties” country=”country”>

Directives have their own scope

= .. birectional binding (documentation)