Intro to AngularJS:

* Include the Angular JS code:
  + <script = “scripts/angular.min.js”></script>
* HTML custom attributes:
  + Data-\* attributes(Bootstrap/jQuery)
  + Angular: ng-\* attributes / data-ng-\*
    - Ex: ng-app, ng-bind, ng-model, ng-init, ng-repeat
* Ng-\* Attributes:
  + Ng-app directive: applied to specify the root of the application
    - Ex. Applying to html tag
  + ngInit Directive: used to evaluate a expression, initialize a javascript variable
  + NgModel Directive: binds the input value to a variable within the scop: Two=way data binding
  + ngRepeat Directive: looping construct
    - loops over items in a collection
    - instantiates a template for each item
* Angular Expressions
  + Simple javascript expressions are evaluated against an Angular scope object
  + No conditionals, no loops or exceptions
  + Expressions enclosed within {{expression}}

Model View Controller Framework

* Software engineering architecture pattern
  + Isolation of domain logic from user interface
  + Permits independent development and testing

Angular Filters

* Built-In Filters:
  + Uppercase/lowercase: converts the text
  + Currency : formats the number as a currency
  + Date: formats the date as per the formatting specified
  + Filter: selects a subset of an array based on the critieria specified and returns a new array
  + Orderby: orders the array based on the criteria specified

Gulp:

* Gulp Streams:
  + gulp.src(): function that takes file globs and creates a stream of objects that represent the files
  + pipe(): allows the stream to be piped through a function
  + gulp.dest(): specifies the destination of the changed files
* Defining the JSHint Task:
  + gulp.task(‘jshint’, function(){

gulp.src(‘app/scripts/\*\*/\*.js’)

.pipe(jshint())

.pipe(jshint.reporter(stylish));

});

* Imagemin
  + gulp.task(‘imagemin;, function() {

return gulp.src(‘app/images/\*\*/\*’)

.pipe(cache(imagemin({optimizationLevel: 3,

progressive:true, interlaced: true})))

.pipe(gulp.dest(‘dist/images’))

.pipe(notify({message: ‘Images task complete’}));

});

* Cleaning up:
  + Makes use of the Node module named del:
    - gulp.task(‘clean’, function() {

return del([‘dist’]);

};

* Usemin:
  + [] mean that before executing usemin, execute jshint
  + Gulp.task(‘usemin’,[‘jshint’], function(){

return gulp.src(‘./app/menu.html’)

.pipe(usemin({

css:[minifycss(),rev()],

js: [uglify(), rev()}))

.pipe(gulp.dest(‘dist/’));

});

* BrowserSync
  + Gulp.task(‘browser-sync’, [‘default’], function() {

var files = [‘app/\*\*/\*.html’, ‘app/styles/\*\*/\*.css’, ‘app/images/\*\*/\*.png’,

‘app/scripts/\*\*/\*.js’, ‘dist/\*\*/\*’ ];

browserSync.init(files { server: {baseDir: “dist”, index: “menu.html”}

})

//Watch any files in dist/, reload on change

gulp.watch([dist/\*\*’]).on(‘change’, browserSync.reload);

};

* Watch keeps a watch on the files:
  + gulp.task(‘watch’, [‘browser-sync’], function(){

//if any scripts/styles change, rerun usemin task

gulp.watch(‘{app/scripts/\*\*/\*.js, app/styles/\*\*/\*.css, app/\*\*/\*.html}’,

[‘usemin’]);

gulp.watch(‘app/images/\*\*/\*’, [imagemin])

};

* Default task:
  + Gulp.task(‘default, [‘clean’], function(){

Gulp.start(‘usemin’, ‘imagemin’, ‘copyfonts’);

});

Angular scope:

* Scope is an object referring to the application model, gluing the view and the controller
  + The controller can set properties on the scope
  + The view binds to the properties set by the controller
  + Angular is responsible to keep two in sync
* $rootScope: the topmost scope, created by Angular when your app starts
  + As Angular traverses the DOM, it creates new scopes when it encounters some directives, creating it as a child of a parent scope:
    - Child has access properties in the parent’s scope
    - Ng-controller, for example, creates a new scope under the root scope
    - Scope tree similar to DOM
* Forms and two-way binding:
  + Define a JS object on the $scope
    - .controller(‘Contact Controller’, [‘$scope’, function($scope) {

$scope.feedback = {mychannel: “”, firstName: “”, lastName: “”, agree:false, email:””};

* + Use ng-model directive on form fields:
    - <input type = “text” class=”form-control id = “firstname” name = “firstname” placeholder = “Enter First Name” ng-model = “feedback.firstName” required>
* Binding Select:
  + Select items acan be bound by defining JS object
    - Var channels = [{value:”tel”, label : “Tel.”},

{value: “Email”, label: “Email”}];

* + Then bind using ng-options directive
    - <select class = “form control” ng-model = “feedback.mychannel”

ng-options = “channel.value as channel.label for channel in channels”>

<option value = “”> Tel or Email?</option>

</select>

* Angular Form Validation
  + Turn off HTML form validation:
    - <form class = “form-horizontal” name = “feedbackForm”

ng-submit=”sendFeedback()” novalidate>

* Angular Form validation directives:
  + Angular validates the form fields before copying the value over to the $scope
    - Ex: ng-minlength, ng-maxlength, ng-pattern: can use regex
* Field Properties:
  + Use Field name and form name:
    - Ex., feedbackForm.firstName.$pristine in the form, and $scope. feedbackForm.firstName.$pristine in the controller code
      * $pristine: true if form has not been changed
      * $dirty: reverse of $pristine
      * $valid: true if form field/whole form is valid
      * $invalid reverse of $valid
* Field Properties and CSS
  + Bootstrap provides a lot of CSS classes to enable display of form validation state and messages:
    - .has-error, .has-warning, .has-success
    - .help-block to display helpful messages below the field
* Dependency Annotation in Angular
  + Inline array annotation
    - Module.controller(‘MenuController’, [‘$scope’, ‘menuFactory’, function($scope, menuFactory) {

}]);

* + $inject property annotation
    - var MenuController = function($scope, menuFactory){

};

* + Implicit annotation
    - Module.controller(‘MenuController’, function($scope, menuFactory) {

}]);

Angular Factory and Service

* Angular’s built in services
  + Built-in services always start with $
    - Ex. $http, $scope, $rootScope, $location, $parse, $templateCache, $animate, $injector
    - Inject them using DI
  + Declare services:
    - Service()
    - Factory()
    - Provider()
    - Constant()
    - Value()
* Angular Factory
  + Angular.module(‘confusionApp’)

.factory(‘menuFactory’, function(){

var menufac = {};

var dishes = […];

menufac.getDishes=function(){

return dishes;

};

menufac.getDish = function(index)

{

return dishes[index];

};

return menufac;

});

* Angular Templaces
  + Angular templates are written w/ HTML
    - Contains angular specific elements and attributes
    - Dynamic View = Template + controller + model
  + Angular elements and attributes:
    - Directives, markup; {{expressions}}, filter, form controls
  + ngInclude Directive:
    - Directive used to fetch, compile, and include an external HTML element
    - Usage:
      * <div ng-include =”menu.html”></div>
      * <ng-include src = “menu.html”></ng-include>
    - creates a new scope

Angular and SPA

* Role of Client and Server in SPA
  + Server: serves up data REST API
    - Supplies the static HTML pages, Angular templates and resources
  + Rendering of view is completely on the client-side
    - Templating and routing(moving from one template to another)
* Deep-Linking: hyperlink that specifies a link to a searchable or indexed piece of web content
  + Any change to the hash portion doesn’t cause a page reload
* The $location Service:
  + Exposes the current URL in the browser address bar:
    - Watch and observe the URL
    - Change the URL
  + Synchronizes the URL with the browser when the user:
    - Changes the address bar
    - Clicks the back/forward buttons
    - Clicks on a link
  + Allows you to manipulate the hash portion of a URL:
    - url(): get/set the URL
    - path(): get/set the path
    - search(): get/set the search part
    - hash(): get/set the hash part
* Routing:
  + Mapping the path portion of a URL to a handler for that particular route
    - Route is the hash portion of the URL in the context of SPAs
    - Example: <http://conFusion.food/index.html#/menu/0>
  + Angular ngRoute Module
    - Installing the module:
      * Bower install angular-route –S
      * Manages the interaction between the $ location service and the rendered view
      * Dependency injection into the module:
        + Angular.module(‘confusionApp;, [‘ngRoute’])
  + The $routeProvider: enabling mapping from the routes to handlers
    - Handlers are object that defines : template URL, controller
  + Angular.module(‘confusionApp’, [ngRoute’]).
  + .config(function($routeProvider){

$routeProvider

.when(‘/contactus, {

templateUrl:’contact.html’, controller: ‘ContactController’

})

.otherwise(‘/contactus’);

* + $routeParams:
    - menu.html:
      * <div class = “media-left media-middle”>
        + <a ng-href = “#/menu/{{dish.\_id}}”>
    - DishDetailController:
      * .controller(‘DishDetailController’, [$scope’, ‘$routeParams’, ‘menuFactory’, function($scope, $routeParams, menuFactory){

var dish = menuFactory.getDish(parseInt($routeParams.id,10));

$scope.dish=dish;

}]);

* + the ngView Directive:
    - directive that works together with $route service to include the rendered template of the current route into the main layout
    - usage:
      * <ng-view></ng-view>
      * <div ng-view></div>

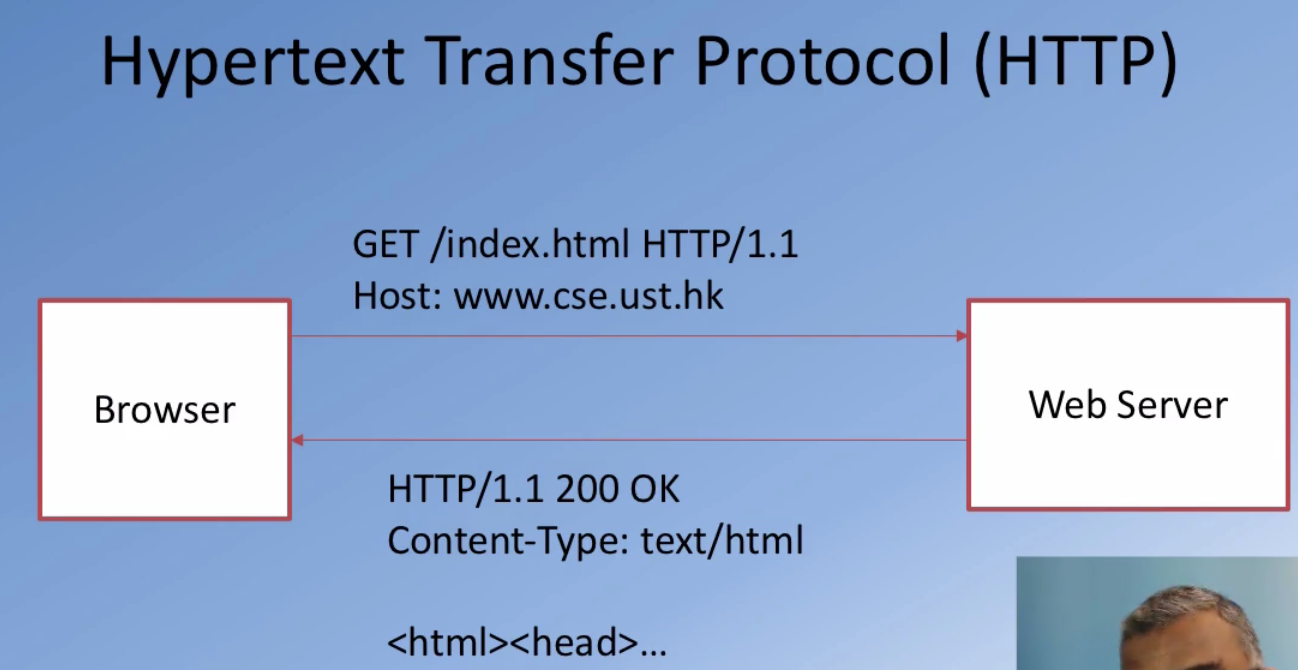
Angular UI-Routers

* Limitations of ngRoute:
  + Only one view allowed per page, no multiple or nested views
  + Application views tied to the route URL
* Angular UI Router:
  + Views based on the state of the app, can change parts of your site using the routing even if the URL doesn’t change, multiple and nexted views
  + Bower install angular-ui-router –S
  + Dependency injection into the module
* uiView Directive
  + Indicates where to include the views
    - <div ui-view = “header”></div>
    - <div ui-view = “content”></div>
    - <div ui-view = “footer></div>
  + ui-sref = “state” to indicate which state to move to when clicked

Client-server communication and Angular Testing

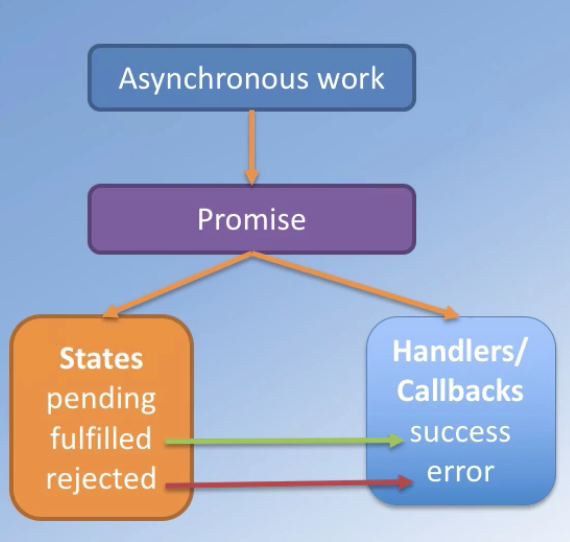
Networking Essentials:

* Web applications are not stand-alone, many of them have a “Cloud” backend
* Hypertext Transfer Protocol: HTTP
  + A client-server communications protocol
  + Allows retrieving interlinked text documents(hypertext)
    - WWW



* HTTP Response:
  + Server may send back data in a specific format:
    - eXtensible Markup Language
    - Javascript Object Notation

Client-Server Communication using $http

* Angular $http
  + $http: Core Angular service to communicate w servers using the HTTP protocol via the browser’s XMLHttpRequest and JSONP
  + operation is asynchronous in nature
* Promise
  + Angular $q service: run functions asynchronously and use the return value (or exceptions) when they are done processing
  + 
* The $http service:
  + Returns a promise:
    - $http({method: ‘GET’, url: ‘/dishes’})
      * .then(function(){…}, function(){…});
  + Shortcut methods:
    - $http.get(), $http.put(), $http.post(), $http.delete(), $http.jsonp(), $http.head()..
* HTTP Response:
  + Response:
    - response.data: string/object containing body of the message
    - response.status: status code
    - Response.headers: header info
    - Response.config: config object
    - Response.statusText: HTTP status text of the response
  + The ngIf Directive
    - The ngIf directive can be used to add/remove a portion of the DOM tree based on an expression:
      * <div class = “col-xs-12” ng-if = “!showMenu”>
        + <h3>{{message}}</h3>
* sda
* a
* a
* a
* a
* a
* a