Week 4

coursera.org/learn/single-page-web-apps-with-angularjs/discussions/weeks/4/threads/qeEw4RUaEeeq4gqhD-WaKA

** This is a review from **Lecture 36** until **Lecture 41** (split in 2 parts)*

AngularJS – Routing with ui-router (Part 1/2)

(In-depth Documentation: github.com/angular-ui/ui-router/wiki)

Going through the steps for using some of the basics of ui-router...

0) Search for "angular-ui-router CDN" (content delivey network) to download the "more stable" **0.3.1** version which is the one used in this course (versions starting with 1.x are for Angular 2 and were stil unstable at the time this course was made).

> Use "Save as..." to save **angular-ui-router.js** or **angular-ui.router.min.js** in the app's lib folder.

1) Reference the **ui-router** package: Since ui-router depends on the core AngularJS package, it needs to be referenced (and loaded by the browser) after it (angular.min.js first, and then angular-ui-router.min.js):

```
<script src="lib/angular.min.js"></script>

<script src="lib/angular-ui-router.min.js"></script>
```

2) The ui-router's custom directive **ui-view**: Designate where in the html page to position a placeholder for the interchangeable view (the ui-view). The content of our views (the templates) will be inserted into its body (will be loaded between the opening and closing "ui-view" tags):

```
<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" href="css/styles.css">

    <title>Menu Items</title>

  </head>

  <body>

    <h1>Welcome to our restaurant!</h1>

    <ui-view></ui-view>

    <script src="lib/angular.min.js"></script>

    <script src="lib/angular-ui-router.min.js"></script>

    <script src="src/app.module.js"></script>

    <script src="src/routes.js"></script>

    <script src="src/menu.component.js"></script>

    <script src="src/menu.controller.js"></script>

    <script src="src/menu.service.js"></script>

  </body>

</html>
```

3) Declare **ui-router** as a dependency by injecting its "**ui.router**" module into our 'App' module (notice it has a dot replacing the dash in its name):

```
(function () {

'use strict';

  angular.module('App', ['ui.router']);

})();
```

4) Config the **ui-router** services: ui-router handles **ui-state** AND **routing through url** as two indepentent functions within two essential services (**$state** and **$urlRouter**) which are configured in the **.config** method by injecting their provider name versions (service name + word "Provider"):

```
(function () {

'use strict';

  angular.module('App).config(RoutesConfig);

  RoutesConfig.$inject = ['$stateProvider', '$urlRouterProvider'];

  function RoutesConfig($stateProvider, $urlRouterProvider) {

    ...

  }

})();
```

5) Set up a default state url: If no states are recognized from the url (no states are yet associated with it), $urlRouterProvider.**otherwise()** method sets a default one:

```
...

  RoutesConfig.$inject = ['$stateProvider', '$urlRouterProvider'];

  function RoutesConfig($stateProvider, $urlRouterProvider) {

    $urlRouterProvider.otherwise('/');

  }

...
```

6) Set up UI states with $stateProvider.**state()** method:

```
...
  RoutesConfig.$inject = ['$stateProvider', '$urlRouterProvider'];
  function RoutesConfig($stateProvider, $urlRouterProvider) {
    $urlRouterProvider.otherwise('/');
    $stateProvider
      .state('home', {
        url: '/',
        templateUrl: 'src/app/templates/home.html'
      })
      .state('menu', {
        url: '/menu',
        templateUrl: 'src/app/templates/menu.html'
      });
  }
...
```

P.S.: Removing url property from a state configuration stops the user from changing states through urls but the changes are still switchable (they're still clickable) through the use of **ui-sref** directive (coming next).

7) Create links and actions for going from one **state** to another. Set links and actions ("anchors", "buttons", etc.) for the views to be opened within the previously set **ui-view** directive. Use **ui-sref** to dynamically switch between states.

a) Create *home.html* template:

```
<a ui-sref="menu">Menu</a>
```

b) Create *menu.html* template with **menuItems component**:

```
<div id="menu" ng-controller='MenuController as menu'>
  <a ui-sref="home">Home</a> &lt; <span>Menu</span>
  <h3>Items</<h3>
  <menu-items items="menu.items"></menu-items>
</div>
```

c) Since we are already there, create *menu.component.html*:

```
(function () {
'use strict';
  angular.module('App')
    .component('menuItems', {
      templateUrl: 'src/app/templates/items.html',
      bindings: {
        items: '<'
      }
    });
})();
```

d) Along with its *items.html* template:

```
<ul>
  <li ng-repeat="item in $ctrl.items">
    {{ item.qtty }} of {{ item.name }}
  </li>
</u>
```

e) And... *menu.service.html*, why not (for testing purposes):

```
(function () {
'use strict';
  angular.module('App')
    .service('MenuService', MenuService);
  MenuService.$inject = ['$q', '$timeout']
  function MenuService($q, $timeout) {
    var service = this;
    var items = [];
    items.push({  name: "Steak", qtty: "450g", desc: "Sirloin steak"
        });
    items.push({  name: "Rice", qtty: "1 portion", desc: "Biryani
        rice" });
    items.push({  name: "Salad", qtty: "Half portion", desc: "Stupid
        salad" });
    service.getItems = function () {
      var deferred = $q.defer();
      $timeout(function () {
        deferred.resolve(items);
      }, 500);
      return deferred.promise;
    };
  }
})();
```

8) Set up **ui-sref-active** to give CSS style for the active state. This will tell which state is active in the html page.

a) UPDATE FILE: *src/app/templates/home.html*

```
<a ui-sref="menu" ui-sref-active="activeState">Menu</a>
```

b) UPDATE FILE: *src/app/templates/menu.html*

```
<div id="menu" ng-controller='MenuController as menu'>

  <a ui-sref="home" ui-sref-active="activeState">Home</a> &lt; <span

      >Menu</span>

  <h3>Items</<h3>

  <menu-items items="menu.items"></menu-items>

</div>
```

c) And configure CSS class ".activeState" in the *style.css* file:

```
.activeState {

  font-weight: bold;

}
```

*** End of Part 1 of 2 ***