Question: 1.

Answer 1: Currently I am exploring angular 2 so there might be mistake to convert given code in angular 2 from angular 1. Since angular is ES6 based therefore need to have well understanding of the type script, which exploring too. Thanks.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Define the test appCodeName

import { Component } from '@angular/core';

@Component({

selector: 'my-app',

templateUrl: 'components/accounts/templates/sample.tpl.html'

})

export class AppComponent { },

//Define the directive as given in code

import { Directive, ElementRef, Input } from '@angular/core';

@Directive({ selector: '[sampleDirective]' })

export class sampleDirective {

constructor(el: ElementRef) {

this.chooseStartDate = function () {

var options = {

date: new Date(),

mode: 'date', // or 'time'

minDate: new Date() - 10000,

allowOldDates: true,

allowFutureDates: false,

doneButtonLabel: 'Next',

doneButtonColor: '#1790C6',

cancelButtonLabel: 'X',

cancelButtonColor: '#333132'

};

}

this.chooseEndDate = function () {

var options = {

date: new Date(),

mode: 'date', // or 'time'

minDate: new Date() - 10000,

allowOldDates: true,

allowFutureDates: false,

doneButtonLabel: 'Done',

doneButtonColor: '#1790C6',

cancelButtonLabel: 'X',

cancelButtonColor: '#333132'

};

};

this.chooseAmount = function () {

var options = {

title: 'Enter Amount',

mode: 'amount',

};

};

this.chooseAmountRange = function () {

var options = {

title: 'Enter Amount Range',

mode: 'amount\_range',

};

};

this.chooseCheck = function () {

var options = {

title: 'Enter Check #',

mode: 'check',

};

};

this.chooseCheckRange = function () {

var options = {

title: 'Enter Check Range',

mode: 'check\_range',

};

};

this.reset = function () {

this.startDate = null;

this.endDate = null;

this.singleAmount = null;

this.minAmountRange = null;

this.maxAmountRange = null;

this.singleCheckNumber = null;

this.minCheckNumber = null;

this.maxCheckNumber = null;

}

}

}

//Main module go here

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

import { sampleDirective } from './sampleDirective.directive';

@NgModule({

imports: [ BrowserModule ],

declarations: [

AppComponent,

sampleDirective

],

bootstrap: [ AppComponent ]

})

export class AppModule { }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Question: 2: Given a standard AngularJS application. Can you explain to me how you would go about appending a piece of data to every http (ajax) call and how you would go about logging every instance of an http (ajax) error?

Answer 2:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Here we have some idea to how do append data in AJAX call using jquery:

a).

(function ($) {

var \_ajax = $.ajax,

A = $.ajax = function(options) {

if (A.data)

if(options.data) {

if(typeof options.data !== 'string')

options.data = $.param(options.data);

if(typeof A.data !== 'string')

A.data = $.param(A.data);

// we can append the data here in each ajax call

options.data += '&' + A.data;

} else

options.data = A.data;

return \_ajax(options);

};

})(jQuery);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

b) . Here are some code which log every instance of http AJAX error,

Idealy we should use try and catch block to trace the error,

'use strict';

var loggingModule = angular.module('loggingModule', []);

/\*\*

\* Service that gives us a nice Angular-esque wrapper around the

\* stackTrace.js pintStackTrace() method.

\*/

loggingModule.factory(

"traceService",

function(){

return({

print: printStackTrace

});

}

);

/\*\*

\* Override Angular's built in exception handler, and tell it to

\* use our new exceptionLoggingService which is defined below

\*/

loggingModule.provider(

"$exceptionHandler",{

$get: function(exceptionLoggingService){

return(exceptionLoggingService);

}

}

);

/\*\*

\* Exception Logging Service, currently only used by the $exceptionHandler

\* it preserves the default behaviour ( logging to the console) but

\* also posts the error server side after generating a stacktrace.

\*/

loggingModule.factory(

"exceptionLoggingService",

["$log","$window", "traceService",

function($log, $window, traceService){

function error(exception, cause){

// preserve the default behaviour which will log the error

// to the console, and allow the application to continue running.

$log.error.apply($log, arguments);

// now try to log the error to the server side.

try{

var errorMessage = exception.toString();

// use our traceService to generate a stack trace

var stackTrace = traceService.print({e: exception});

// use AJAX (in this example jQuery) and NOT

// an angular service such as $http

$.ajax({

type: "POST",

url: "/logger",

contentType: "application/json",

data: angular.toJson({

url: $window.location.href,

message: errorMessage,

type: "exception",

stackTrace: stackTrace,

cause: ( cause || "")

})

});

} catch (loggingError){

$log.warn("Error server-side logging failed");

$log.log(loggingError);

}

}

return(error);

}]

);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Question : 3 write some unit tests, using using your framework of choice, to fully test the code

Answer 3 {

// Test cases using Jasmine for given code in .md file

(function () {

"use strict";

describe("sampleCtrl", function () {

/\* global $q, $scope \*/

var isSampleCtrl, isSampleHttp, $http;

beforeEach(function () {

inject(function (\_$injector\_) {

$http = \_$injector\_.get("$http");

});

isSampleCtrl = sampleCtrl();

isSampleCtrl.name = 'Eastern Labs';

isSampleCtrl.counter = 0;

spyOn($scope, "$watch").and.callThrough();

isSampleHttp = sampleHttp();

$httpBackend.whenGET("apiURL").respond({

result: {

resultId: "12345",

resultCode: "0",

resultText: "Success"

}

});

spyOn($http, "getHttp").and.callFake(function (c) {

return $q(function (resolve) {

resolve();

});

});

});

// test to sampleCtrl

it("should watch the scope for name changes", function() {

expect($scope.$watch).toHaveBeenCalledWith(counter, 1);

});

// test to sampleHttp

it("should call the rest api", function () {

isSampleHttp.getHttp();

expect(isSampleHttp.getHttp).toHaveBeenCalled();

}

function sampleCtrl() {

return $controller("sampleCtrl", {

$scope: $scope

});

}

function sampleHttp() {

return $controller("sampleCtrl", {

$scope: $scope

});

}

});

}());

}

Question: 4. Write code to make this work:

[1,2,3,4,5].duplicate(); // [1,2,3,4,5,1,2,3,4,5]

Answer: 4 Writing prototype on Array:

code:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

There are two approcah to do this, we can follow any one

a).

Array.prototype.duplicate = function() {

var arrLen = this.length;

for (var i = 0; i < arrLen; i++) {

this[arrLen + i] = this[i];

}

return this;

}

b)

Array.prototype.duplicate = function () {

var array = this;

return array.concat(array);

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/