**Question 1**

The task relates to fetch and display the code and title by making an ajax call to a WebApi using Angular JS. Fill in the appropriate code to complete the below code base. (Feel free to copy the code below, modify it, and email your solution)

<body data-ng-app="app">

<div ng-controller=“TitleController">

<button id="get-items-button" ng-click="getTitles()">Get Pets</button>

<p>Look at the list of titles!</p>

<!--this table shows the titles we get from our service-->

<table cellpadding="0" cellspacing="0">

<thead> <tr> <th>Code</th> <th>Title</th> </tr> </thead>

<tbody>

**<tr** ng-repeat=”title in titles”**>**

<td>{{title.titleCode}}</td>

<td>{{title.title}}</td>

</tr>

</tbody>

</table>

</div>

<script>

(function () {

//create the module

angular.module('app', [])

//add controller

.controller(‘TitleController', function (($scope, $http) {

//declare an array of items. this will get populated with our ajax call

$scope.titles = [];

//declare an action for our button

$scope. getTitles = function () {

var url = “/api/Codes/Titles/”;

//perform ajax call and on success copy the data we get into the items array so that angular can track the object and bind it automatically. On error alert the message “Error getting data”.

$http({method : "GET", url : url})

.then(function mySuccess(response) {

$scope.titles = response.data;

}, function myError(response) {

alert(“Error getting data”)

});

} });

})();

</script>

**Question 2:**

1. To set up a decent testing environment for your AngularJS application, you will need several npm modules. Provide some examples.

* jasmine-core, karma, karma-chai, karma-chrome-launcher, karma-cli, karma-jasmine, karma-webpack

1. Filters, services and factories can be compared to Inversion of Control/Dependency Inversion in a traditional object-oriented framework.
2. What is great about AngularJS’ way of injecting dependencies?

I don’t know have experience in Angular 1 but in Angular 2+ Reasons are:

* Reducing the dependency to each other of objects in application.
* Loosely couple or it promotes decoupling to application.
* Because they are Loosely couple we can now easily test the application objects.
* Re-usability of code or objects in different applications.

1. What will be the approach to implement a directive for the codebase task in Question 1?

I don’t know have experience in Angular 1 but in Angular 2+

To consume

<App-Title></App-Title>

Title Component template (title.component.html')

<table cellpadding="0" cellspacing="0">

<thead> <tr> <th>Code</th> <th>Title</th> </tr> </thead>

<tbody>

**<tr** \*ngFor="let title of titles;”**>**

<td>{{title.titleCode}}</td>

<td>{{title.title}}</td>

</tr>

</tbody>

</table>

</div>

TypeScript Component Directive

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

import { Http, Headers,RequestOptions } from '@angular/http';

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

@Component({

moduleId: module.id.toString(),

selector: 'App-Title',

templateUrl: './title.component.html'

})

export class TitleComponent implements OnInit {

public titles: any = []

private \_baseUrl = ''

constructor(private http: Http, @Inject('BASE\_URL') baseUrl: string) {

this.\_baseUrl = baseUrl;

}

ngOnInit() {

this.http

.get(this.\_baseUrl + 'api/titles').subscribe(result => {

this.titles = result.json() as any;

}, error => console.error(error));}

}

Make sure all compoenents are add from the NGModule

import { TitleComponent } from './title.component;

@NgModule({

declarations: [TitleComponent],

……