Apex Institute of Technology

Program Name: BE CSE (Cloud Computing)

LAB MANUAL

Semester	:	7th
Course Name	:	Web Frameworks in Cloud: AngularJS
Course Code	:	CSP-441
Course Coordinator	:	Dr. Mohammad Junedul Haque

Vision of the University

“To be globally recognized as a Centre of Excellence for Research, Innovation, Entrepreneurship and disseminating knowledge by providing inspirational learning to produce professional leaders for serving the society.”

Mission of the University

M1: Providing world class infrastructure, renowned academicians and ideal environment for Research, Innovation, Consultancy and Entrepreneurship relevant to the society.

M2: Offering programs & courses in consonance with National policies for nation building and meeting global challenges.

M3: Designing Curriculum to match international standards, needs of Industry, civil society and for inculcation of traits of Creative Thinking and Critical Analysis as well as Human and Ethical values.

M4: Ensuring students delight by meeting their aspirations through blended learning, corporate mentoring, professional grooming, flexible curriculum and healthy atmosphere based on co-curricular and extra-curricular activities.

M5: Creating a scientific, transparent and objective examination/evaluation system to ensure an ideal certification.

M6: Establishing strategic relationships with leading National and International corporates and universities for academic as well as research collaborations.

M7: Contributing for creation of healthy, vibrant and sustainable society by involving in Institutional Social Responsibility (ISR) activities like rural development, welfare of senior citizens, women empowerment, community service, health and hygiene awareness and environmental protection.

Vision of the Department

To be recognized as a centre of excellence for Computer Science & Engineering education and research, through effective teaching practices, hands-on training on cutting edge computing technologies and excellence in innovation, for creating globally aware competent professionals with strong work ethics whom would be proficient in implementing modern technology solutions and shall have entrepreneurial zeal to solve problems of organizations and society at large.

Mission of the Department

M1: To provide relevant, rigorous and contemporary curriculum and aligned assessment system to ensure effective learning outcomes for engineering technologies.

M2: To provide platform for industry engagement aimed at providing hands-on training on advanced technological and business skills to our students.

M3: To provide opportunities for collaborative, interdisciplinary and cutting-edge research aimed at developing solutions to real life problems

M4: To imbibe quest for innovation, continuous learning and zeal to pursue excellence through hard work and problem-solving approach

M5: To foster skills of leadership, management, communication, team spirit and strong professional ethics in all academic and societal endeavors of our students

Programme Educational Objectives, Programme Specific Outcomes and Programme Outcomes

Programme Educational Objectives (PEOs). The PEOs are broad statements that describe the career and professional accomplishments that the program is preparing its graduates to achieve in few years (for example three years) subsequent to receiving the degree. The PEOs of the B.E. program in Computer Science and Engineering are as follows:

PEO 1 CU Computer Science and Engineering graduates will have a bright career as Cloud Developer, Cloud Architect, Cloud Analyst, Researcher, or Entrepreneur and be able to adapt to the evolving technical challenges and the changing career opportunities. Also, will be learn to effectively communicate ideas in oral written or graphical form and to promote collaboration with other members of engineering teams.

PEO 2 CU Computer Science and Engineering graduates will analyze a problem, identify and define the computing requirements appropriate to its solution in addition to the ability to design, implement and evaluate a computer-based system, process, component, or programs to meet desired needs. They will be able to continue to demonstrate the professional skills and communicative abilities necessary to be competent employees, assume leadership roles and have career success and satisfaction.

PEO 3 CU Computer Science and Engineering graduates will function effectively as a member or a leader of a diverse team under different environments and multidisciplinary work can be carried by the team.

PEO 4 CU Computer Science and Engineering graduates will have the ability to readily adapt to changing environments by continuously learning new state of the art technologies.

PEO 5 CU Computer Science and Engineering graduates will inculcate characteristics needed for leadership roles, professional ethics, excellence and active participation in a successful career while working in an interdisciplinary team.

Programme Specific Objectives (PSOs) are specific statements that describe the professional career accomplishments that the program is designed. The PSOs of the B.E. program in Computer Science and Engineering are as follows:

PSO 1 CU Computer Science and Engineering graduates will be able to develop and deploy cloud application using popular cloud platforms.

PSO 2 CU CSE graduates will be able to compare, contrast, and evaluate the key trade-offs between multiple approaches to cloud system design, and identify appropriate design choices when solving real-world cloud computing problems.

PSO 3: CU CSE graduates will be able to make recommendations on cloud computing solutions for an enterprise.

Programme Outcomes (POs) are attributes of the graduates of the programme that are indicative of the graduates' ability and competence to work as an engineering professional upon graduation. Program Outcomes are statements that describe what students are expected to know or be able to do by the time of graduation. They must relate to knowledge and skills that the students acquire from the programme. The achievement of all outcomes indicates that the student is well prepared to achieve the program educational objectives down the road. The following 12 POs have been chosen by the Computer Science and Engineering Department of Chandigarh University. The Computer Science and Engineering curriculum at CU has been designed to fully meet all the 12 Programme Outcomes:

PO 1 An ability to apply knowledge of science, mathematical foundations, algorithmic principles, and computer science and engineering theorems in the modelling and design of computer-based systems to real-world problems (Engineering Knowledge)

PO 2 Honed skill set comprising of evolutionary tools and techniques applicable in computing & engineering practice (Problem analysis)

PO 3 To recognize, understand the real-world problems and to deploy and execute engineering solutions for sustainable eco system (Design/development of solutions)

PO 4 Ability to harness design and development principles in the construction of verifying complex hardware and software systems (Conduct investigations of complex problems)

PO 5 An ability to come forward with lateral & efficient solutions using modern tools for computing & engineering-based problems (Modern tool usage)

PO 6 An educational ideology to assess the local & global impact of computing and engineering solutions on individuals, organizations and society (The engineer and society)

PO 7 An ability to develop a computer-based system, process, component, or program to meet the desired needs, within realistic constraints such as economic, environmental, social, political, health and safety, manufacturability, and sustainability (Environment and sustainability)

PO 8 An ability to challenge and solve professional, interpersonal, ethical, security & social issues along with ability to perform responsibilities & duties (Ethics)

PO 9 An ability to participate effectively in multi-disciplinary teams (Individual and team work)

PO 10 The ability to work in Professional environment and to communicate effectively with peers (Communication)

PO 11 Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments (Project management and finance)

PO 12 The knowledge by recognizing professional development by pursuing post graduate studies or face competitive examinations that offer challenging and rewarding careers in computing (Life-long Learning).

Scheme Version: 2022	**Name of Course:** Web Framework in Cloud(AngularJS) Lab (CSP-441)	**L**	**T**	**P**	**S**	**C**
	Programs: BE-CSE (CC)	**-**	**-**	**4**	**-**	**1**

Total Marks: 100 **Internal Marks: 40** **External Marks: 60**		
	Pre-requisite: Knowledge of web development	**Total hours =30**
	Course Objective	
1	Develop understanding among the students about the different web frameworks available in cloud, architecture of cloud and REST API's	
2	Demonstrate methods for cloud adoption to business to build business applications using AngularJS	
3	Teach use and application of AngularJS tools and techniques in real case scenarios	

Course Outcomes	
1	The students will develop a project with the concepts cloud web development, REST API's, use of REST API's and implement the same using AngularJS framework
2	The students shall become equipped to build AngularJS applications for helping businesses going into digital transformation
3	The students shall be able to compare within different frameworks available and select best possible framework

Lab Experiments with CO Mapping

S.NO.	Experiment	Mapped CO
Unit-I:- Creation of AngularJS project		
1	Experiment 1.1 Creating a New Project and understanding the libraries	CO1
2	Experiment 1.2 Running Your Project and pushing it to bitbucket	CO1
3	Experiment 1.3 Styles: Using a CSS Framework and inserting it into the component	CO2
Unit 2:- Development of Components		
4	Experiment 2.1 Creating First Component and testing using different frameworks	CO2
5	Experiment 2.2 Creating Data Structures and implementation in AngularJS	CO3
6	Experiment 2.3 Passing data into a component	CO3
7	Experiment 2.4 Looping Over data into AngularJS components	CO3
Unit 3:- Reusable components and formatting data		
8	Experiment 3.1 Formatting data for display within components and interaction between components	CO3
9	Experiment 3.2 Reusability of different components within the application	CO3
10	Experiment 3.3 Responding to an event within the AngularJS framework	CO3

MODE OF EVALUATION: The performance of students is evaluated as follows:

	Practical	
Components	**Continuous Internal Assessment (CAE)**	**Semester End Examination (SEE)**
Marks	60	40
Total Marks	100	

Course Outcome	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O 1	PS O 2	PSO 3
CO1	1	1	2	-	1	-	-	-		-	-	-	2	1	2
CO2	-	1	2	-	2	-	-	-	-	-	-	-	-	1	-
CO3	-	2	-	-	1	-	-	-	-	-	-	-	-	-	-

EXPERIMENT 1.1

Mapped Course Outcomes- CO1

CO1: The students will develop a project with the concepts cloud web development, REST API's, use of REST API's and implement the same using AngularJS framework.

AIM:

Creating a New Project and understanding the libraries.

Objectives

1. Verify the Angular CLI is installed
2. Create a new Angular project
3. Open the new project
4. Review the default project structure

Steps

Verify the Angular CLI is installed

1. Open a command prompt (Windows) or terminal (Mac).

> You can be in *any* directory when you run the command because the Angular CLI is installed globally.

2. Run the command.

```
ng v
```

3. Verify the output.

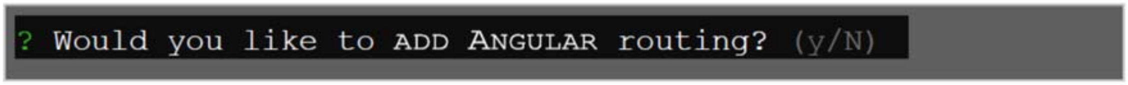
Create a new Angular Project

4. Change the current directory to AngularCourse\code\labs\working.

5. Run the command.

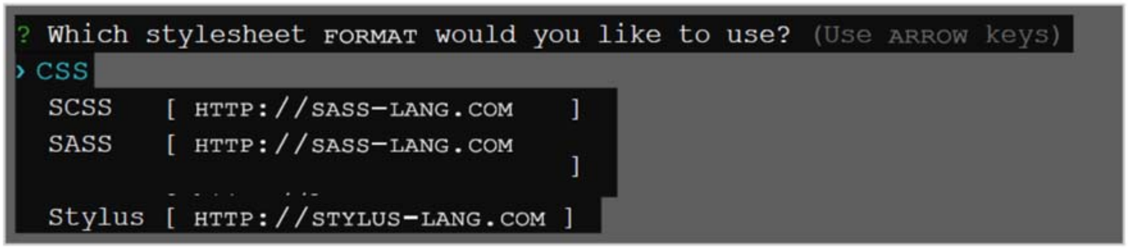
```
ng new PROJECT-MANAGE
```

6. You will receive the following prompt. Type y to answer yes.



```
? Would you like to ADD ANGULAR routing? (y/N)
```

7. You will receive another prompt. Hit enter to accept the default of CSS.



```
? Which stylesheet FORMAT would you like to use? (Use ARROW keys)
› CSS
  SCSS   [ HTTP://SASS-LANG.COM   ]
  SASS   [ HTTP://SASS-LANG.COM
                                  ]
  Stylus [ HTTP://STYLUS-LANG.COM ]
```

8. A new Angular project will be created for you.

> This could take a several minutes and requires an internet connection to install Angular and the other libraries from **npmjs.com**.

> Adding Angular routings tells the Angular CLI to create a routing module where we can configure our routes.
>
> Choosing CSS tells the CLI we want don't want to use a preprocessor for our styles.

Open the new project

9. Change the current directory (cd) to the directory you created in the last step.

```
cd project-manage
```

10. Open the project-manage directory in your editor of choice.

If you are using Visual Studio Code you can run following command:

```
code .
```

.

Review the default project structure

11. Take a few minutes to go over the default project structure with your instructor. Below are some things to discuss.

    a. Open package.json and review the dependencies (JavaScript libraries) installed as well as the scripts.

    b. Understand each of the files involved in bootstrapping (starting) an Angular application.

        1. app.component.html | app.component.ts

        2. index.html

        3. app.module.ts

        4. main.ts

Questions: Viva Voce

Question 1: What is AngularJS?

Question 2: What are the key features of AngularJS?

EXPERIMENT 1.2

Mapped Course Outcomes- CO1

CO1: The students will develop a project with the concepts cloud web development, REST API's, use of REST API's and implement the same using AngularJS framework.

AIM:

Running Your Project and pushing it to bitbucket.

Objectives

1. Run the project
2. Make a change and see the app update

Steps

Run the project

1. If you don't already have one open, open a command prompt (Windows) or terminal (Mac). Set the directory to project-manage.

```
ng serve --open
```

The flag --**open** automatically opens your default web browser with the application running in it.

2. Run the command.

3. The project will:

- build and bundle the code
- open a development web server
- open your default web browser

4. The page should display an Angular logo and the text shown below.

```
Welcome to project-manage!
```

If your default browser is **Internet Explorer** you will see a blank page because the **polyfills** needed to support **IE** are not included by default.

Make a change and see the app update

5. Open and edit the file:

```
src\app\app.component.ts

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})     — —
export class AppComponent {
  ~~title = 'project-manage';~~
```

6. Save your changes.
7. The browser should automatically reload and display.

```
Welcome to my house!
```

Questions: Viva Voce

Question 1: What is Angular Expression? Explain the key difference between angular expressions and JavaScript expressions

Question 2: How can you initialize a select box with options on page load?

EXPERIMENT 1.3

Mapped Course Outcomes-CO2

CO2: The students shall become equipped to build AngularJS applications for helping businesses going into digital transformation.

Aim-

Styles: Using a CSS Framework and inserting it into the component.

Objectives

1. Install a CSS framework
2. Stop and restart the build and web server
3. Verify styles are working in app

Steps

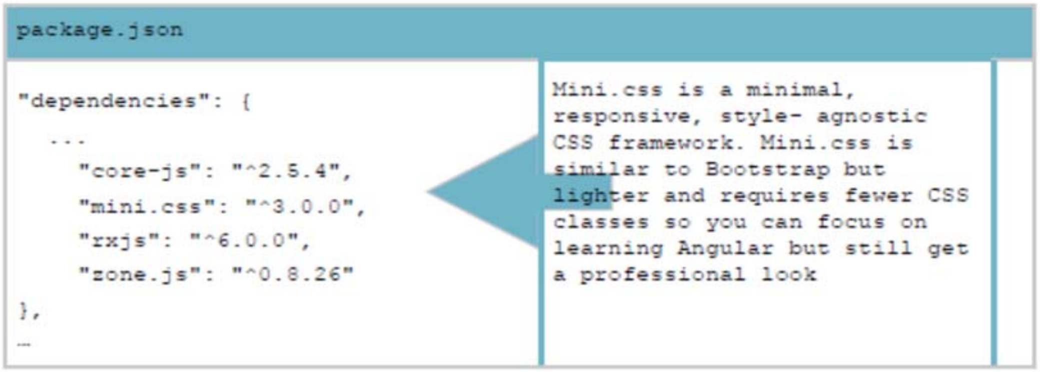
Install a CSS framework

1. Open a a new (leave ng serve running) command prompt (Windows) or terminal (Mac). Set the directory to project-manage.

2. Run the command.

```
npm install mini.css@3.0.0
```

- The JavaScript package manager npm automatically adds mini.css as a dependency.



package.json

```
"dependencies": {
  ...
    "core-js": "^2.5.4",
    "mini.css": "^3.0.0",
    "rxjs": "^6.0.0",
    "zone.js": "^0.8.26"
},
…
```

Mini.css is a minimal, responsive, style- agnostic CSS framework. Mini.css is similar to Bootstrap but lighter and requires fewer CSS classes so you can focus on learning Angular but still get a professional look

3. Include the framework's stylesheet in the Angular CLI's configuration.

In WebStorm files with a *.min.css extension are hidden under the un-minified version of the file.

Stop and restart the build and web server

4. Focus your cursor in the command prompt (Windows) or terminal (Mac). and use [Ctrl+C] to stop the build and web server.

Windows users will be prompted if it is OK to terminate the process and should answer [y+**enter**].

5. Run the command.

6.

```
ng serve --open
```

Your current directory should still be set to **project-manage** or the above command will not work.

7. The application will build and open a browser.

Verify styles are working in app

8. Open the file app\app.component.html.

9. Delete all contents from the file.

10. Add the following quote.

```
src\app\app.component.html
```

```
<blockquote cite="Benjamin Franklin">
    Tell me and I forget, teach me and I may remember, involve me and
    learn.
 </blockquote>
 snippets\lab03-step09.html
```

11. Save your changes.

12. The browser should automatically reload and display

Questions: Viva Voce

Question 1: How Angular JS routes work?

Question 2: Which are the different layers that define cloud architecture?

EXPERIMENT 2.1

Mapped Course Outcomes-CO2

CO2: The students shall become equipped to build AngularJS applications for helping businesses going into digital transformation.

Aim- Creating First Component and testing using different frameworks.

Generate Angular Component using Angular CLI

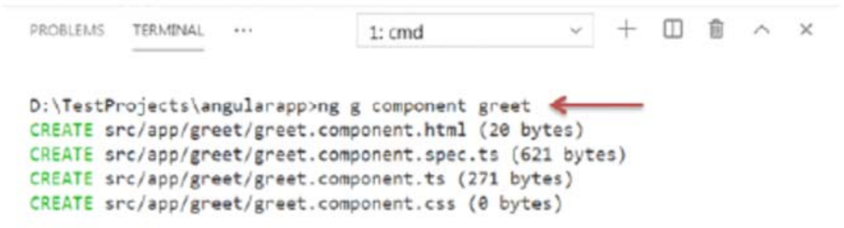
You can create files for a component manually or using the Angular CLI command. Angular CLI reduces the development time. So, let's use Angular CLI to create a new component.

Use the following CLI command to generate a component.

```
ng generate component <component name>
```

All Angular CLI command starts with `ng`, `generate` or `g` is a command, `component` is an argument and then the name of the component.

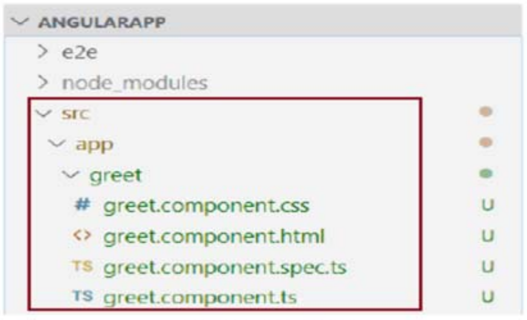
The following executes the `ng g` command to generate the `greet` component in VS Code.



```
D:\TestProjects\angularapp>ng g component greet
CREATE src/app/greet/greet.component.html (20 bytes)
CREATE src/app/greet/greet.component.spec.ts (621 bytes)
CREATE src/app/greet/greet.component.ts (271 bytes)
CREATE src/app/greet/greet.component.css (0 bytes)
```

Create Angular Component

The above command will create a new "greet" folder and app folder and create four files, as shown below.



Component Files

Above, `greet.component.css` is a CSS file for the component, `greet.component.html` is an HTML file for the component where we will write HTML for a component, `greet.component.spec.ts` is a test file where we can write unit tests for a component, and `greet.component.ts` is the class file for a component.

Component Naming Convention

All the component files in Angular should follow the following format:

```
<component-name>.component.<file-type>
```

A component file should include `.component` in name prefixed with the component name and followed by file type. For example, a TypeScript file for our greet component is named `greet.component.ts` . As you have noticed, all the files of greet component are having the same naming conventions. Visit Angular Style guide to know more about it.

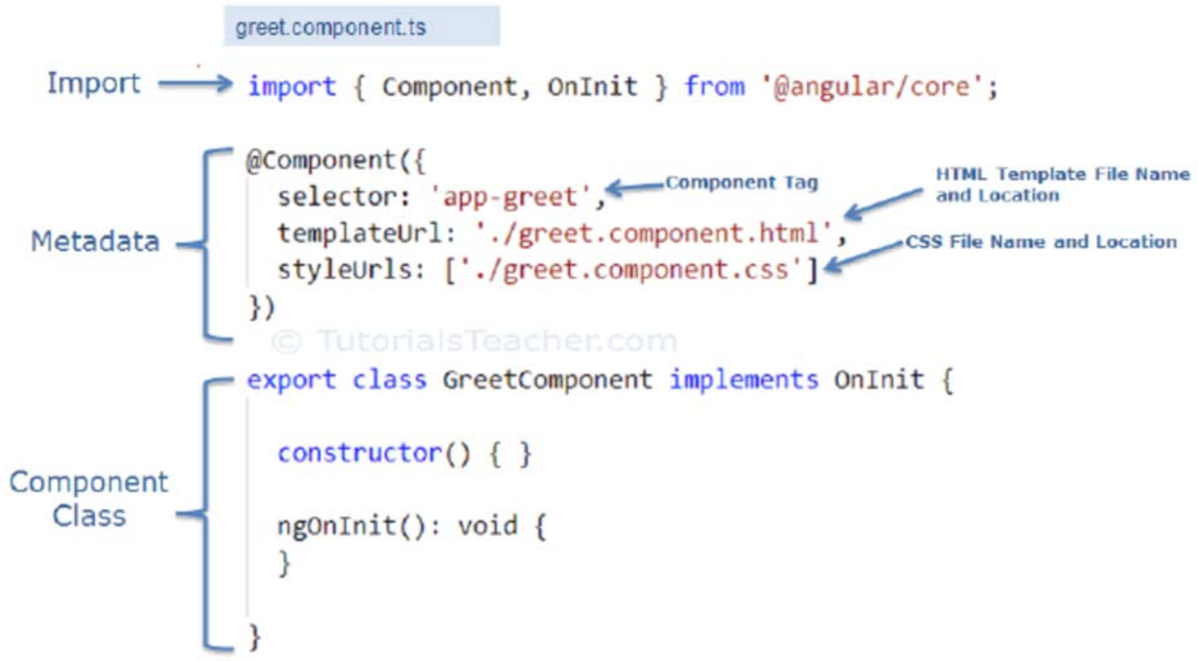
Now, open `greet.component.ts` file in VS Code, and you will see the following code.

Example: Component Class

Copy

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

@Component({
  selector: 'app-greet',
  templateUrl: './greet.component.html',
  styleUrls: ['./greet.component.css']
})
export class GreetComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

The following figure illustrates the important part of the component class.


greet.component.ts

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

@Component({
  selector: 'app-greet',
  templateUrl: './greet.component.html',
  styleUrls: ['./greet.component.css']
})
export class GreetComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

The `greet.component.ts` includes the following parts:

Component Class: `GreetComponent` is the component class. It contains properties and methods to interact with the view through an Angular API. It implements the `OnInit` interface, which is a lifecycle hook.

Component Metadata: The `@Component` is a decorator used to specify the metadata for the component class defined immediately below it. It is a function and can include different configs for the component. It instructs Angular where to get required files for the component, create and render component. All Angular components must have `@Component` decorator above the component class.

The import statement gets the required feature from the Angular or other libraries. Import allows us to use exported members from external modules. For example, `@Component` decorator and `OnInit` interface are contained in `@angular/core` library. So, we can use them after importing it.

Now, let's add a property and method in the component class, as shown below.

Example: Add Properties and Methods in the Component Class

Copy

```
export class GreetComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

  name: string = "Steve";

  greet(): void {
      alert("Hello " + this.name);
  };

}
```

Above, we have added the `name` property and the `greet` method in the component class. Let's use these in the HTML template.

Open `greet.component.html` file, remove existing code and add the following code.

greet.component.ts

Copy

```
<div>
    Enter Your Name: <input type="text" value={{name}} /> <br />
    <button (click)="greet()">Greet Me!</button>
</div>
```

In the above HTML template, we used name property in the {{ }} interpolation to display its value and `greet()` function as click event. Lean more about it in event binding section.

Bootstrapping Component

Now, it's time to load our component, but before that, we need to host our application and load the root component. This process is called bootstrapping.

Angular is a single page application (SPA) framework. So, we need to host our application in `index.html`, and then we need to define a root module to bootstrap our root component. `Index.html` will be the only web page in an Angular application, and that's why it is called SPA.
When you generate Angular application using Angular CLI, it automatically creates `index.html`, root component `app.component.ts`, root module `app.module.ts`, and HTML template `app.component.html` for you. The `AppComponent` class in `app.component.ts` is a root component, and the `AppModule` class in `app.module.ts` is a root module.

1. Declare a component in the root module.

We want to load our new `GreetComponent` into the root component. So, the root module must know about it. We can do it by adding the `GreetComponent` in the declarations array in `app.module.ts` , as shown below.

Example: Add Component in the root module app.module.ts

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { GreetComponent } from './greet/greet.component'; //import GreetComponent

@NgModule({
  declarations: [
    AppComponent,
    GreetComponent  // <- include GreetComponent in declarations
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

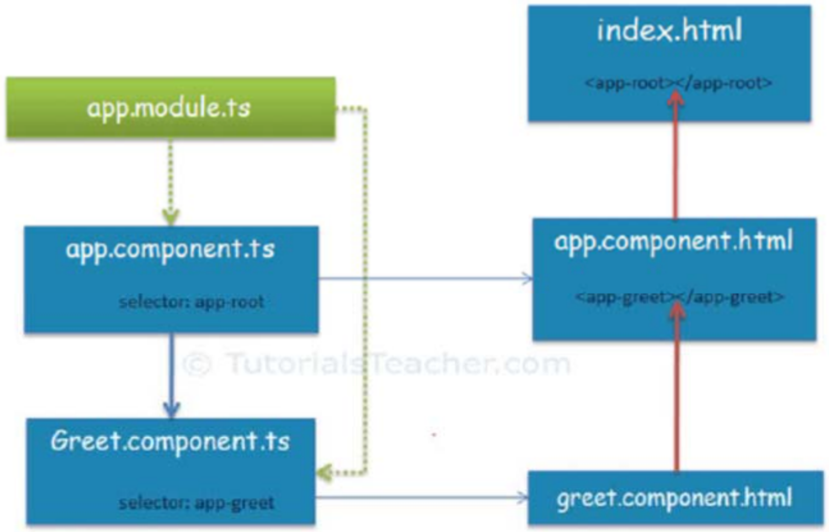
2. Add component tag in the root HTML template.

After adding a component declaration, use component tag `<app-greet></app-greet>` in the HTML file of the root component, which is `app.component.html` , as shown below.

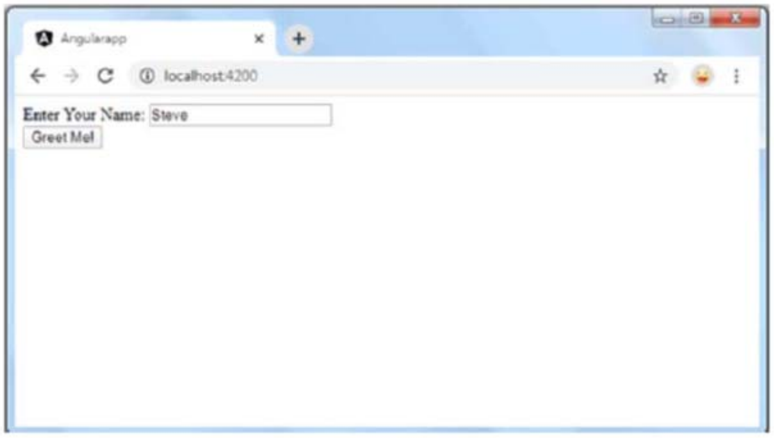
app.component.html

```
<div>
    <app-greet></app-greet>
</div>
```

The following figure illustrates the component bootstrap process.



Run the app by executing npm start command in the terminal of VS Code. After successful compilation, open the browser and enter http://localhost:4200. The following page will be displayed.



This is how we can create and use custom components in Angular.

Questions: Viva Voce

Question 1: What are the benefits of AngularJS?

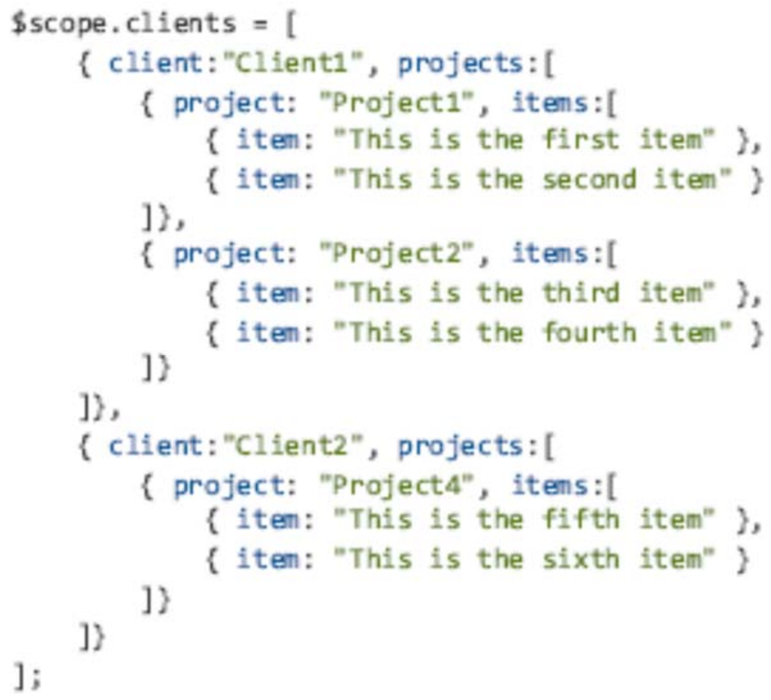
Question 2: What are the steps for the compilation process of HTML?

EXPERIMENT 2.2

Mapped Course Outcomes-CO3

CO3: The students shall be able to compare within different frameworks available and select best possible framework

Aim- Creating Data Structures and implementation in AngularJS



```
$scope.clients = [
    { client:"Client1", projects:[
        { project: "Project1", items:[
            { item: "This is the first item" },
            { item: "This is the second item" }
        ]},
        { project: "Project2", items:[
            { item: "This is the third item" },
            { item: "This is the fourth item" }
        ]}
    ]},
    { client:"Client2", projects:[
        { project: "Project4", items:[
            { item: "This is the fifth item" },
            { item: "This is the sixth item" }
        ]}
    ]}
];
```

Questions: Viva Voce

Question 1: Explain directives and their types

Question 2: Explain injector in AngularJS

EXPERIMENT 2.3

Mapped Course Outcomes-CO3

CO3: The students shall be able to compare within different frameworks available and select best possible framework

Aim- Passing data into a component.

There are two ways to pass data into a component, with 'property binding' and 'event binding'. In Angular, data and event change detection happen top-down from parent to children. However, for Angular events we can use the DOM event mental model where events flow bottom-up from child to parent. So, Angular events can be treated like regular HTML DOM based events when it comes to cancellable event propagation.

The @Input() decorator defines a set of parameters that can be passed down from the component's parent. For example, we can modify the HelloComponent component so that name can be provided by the parent.

```
import { Component, Input } from "@angular/core";

@Component({
  selector: "rio-hello",
  template: "<p>Hello, {{name}}!</p>",
})
export class HelloComponent {
  @Input() name: string;
}
```

The point of making components is not only encapsulation, but also reusability. Inputs allow us to configure a particular instance of a component.
We can now use our component like so:

```
<!-- To bind to a raw string -->
<rio-hello name="World"></rio-hello>
<!-- To bind to a variable in the parent scope -->
<rio-hello [name]="helloName"></rio-hello>
```

Questions: Viva Voce

Question 1: Explain the styling form that ngModel adds to CSS classes

Question 2: What is DI (Dependency Injection) and how an object or function can get a hold of its dependencies?

EXPERIMENT 2.4

Mapped Course Outcomes-CO3

CO3: The students shall be able to compare within different frameworks available and select best possible framework

Aim- Looping Over data into AngularJS components

You can loop through an Array or an Object in AngularJS using the forEach() function. The function invokes the iterator function that iterates or loops through each item in an array.
I am sharing two examples here, showing how to use the forEach() loop in AngularJS to extract items or values from array or an object.

---

forEach() Syntax

```
angular.forEach(object, iterator, [context])
```

object: The object interate over.

iterator: The iterator function or the code.

context: Object to become context (using this) for the iterator function.

Using Angular forEach() with an Object

In the first example, I am creating an object named employees, which has an array of data in it, such as the date of joining, name and age of each employee. I'll use the forEach() function to loop through each value in the object and extract the values and display it (in the console) one by one.

The Script

```
<script>
   var myApp = angular.module('myApp', []);

   myApp.controller('myController', ['$scope', '$filter', function ($scope, $filter) {
        // 'employees' OBJECT, WITH AN ARRAY OF DATA.
        $scope.employees = {
            '05/17/2015': { 'name': 'Alpha', 'age': 37 },
            '06/25/2016': { 'name': 'Bravo', 'age': 27 },
            '09/11/2016': { 'name': 'Charlie', 'age': 29 },
            '01/17/2017': { 'name': 'Delta', 'age': 19 },
            '03/09/2014': { 'name': 'Echo', 'age': 32 }
        }

        // ITERATE THROUGH ITEMS IN OBJECT.
        angular.forEach($scope.employees, function (value, key) {
            var d = new Date('01/05/2016');        // DATE TO COMPARE (A SPECIFIED DATE).
            d = $filter('date')(d, 'shortDate');

            var d1 = new Date();
            d1 = $filter('date')(key, 'shortDate');

            //CHECK IF THE DATE IN OBJECT IS GREATER THAN OR EQUAL TO THE
SPECIFIED DATE.
            if (Date.parse(d1) >= Date.parse(d)) {
                // SHOW THE EXTRACTED DATA.
                console.log(key + ": " + value.name + ": " + value.age);
            }
        });
    } ]
    );
</script>
```

The forEach() function will loop through each value in the object. Here the key has the date and value has name and age. Inside the loop, I am checking a condition that compares the date in the object with the specified date.

Using Angular forEach() on a JSON Array

This is the 2nd example. You can apply forEach() function on a JSON Array too. In this example, I have a list of computer accessories in JSON array format. Using forEach(), the function will loop through each item and bind it to the View, using an Expression ({{ }}).

The View and the Controller

```
<!DOCTYPE html>
<html>
<head>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.5/angular.min.js"></script>
</head>

<body>
   <div ng-app="myApp"
      ng-controller="myController">

      <p><button ng-click="submit(list)">Show Items</button></p>
      <p>{{the_list}}</p>
   </div>
</body>

<script>
   var myApp = angular.module('myApp', []);
   myApp.controller('myController', ['$scope', function ($form) {

      $form.list = [
         { name: 'Keyboard' },
         { name: 'Speakers' },
         { name: 'Mice' },
         { name: 'Routers and USBs' },
         { name: 'Monitors'}];

      $form.submit = function (list) {

        var myAccess = [];

        angular.forEach(list, function (value, key) {
            myAccess.push(value.name);
        });

        if (myAccess.length > 0)
            $form.the_list = myAccess.toString();
    }
  } ]
  );
</script>
</html>
```

In the above example, I am using only one parameter of the forEach() function that is value. I am pushing the values one by one into an array named myAccess (or myAccessories). Finally, I'll bind the list of accessories to my View using the Expression (the_list).

Questions: Viva Voce

Question 1: Explain the concept of scope hierarchy

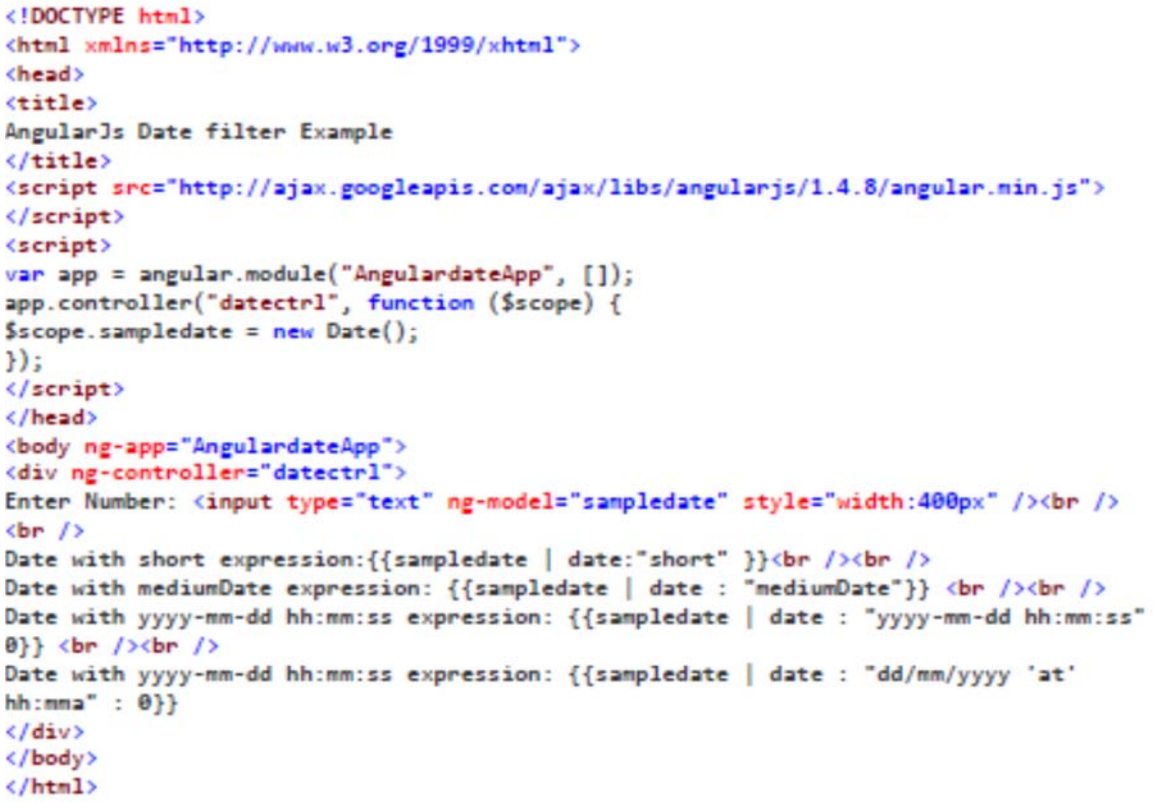
Question 2: What do you mean by cloud delivery models?

EXPERIMENT 3.1

Mapped Course Outcomes-CO3

CO3: The students shall be able to compare within different frameworks available and select best possible framework

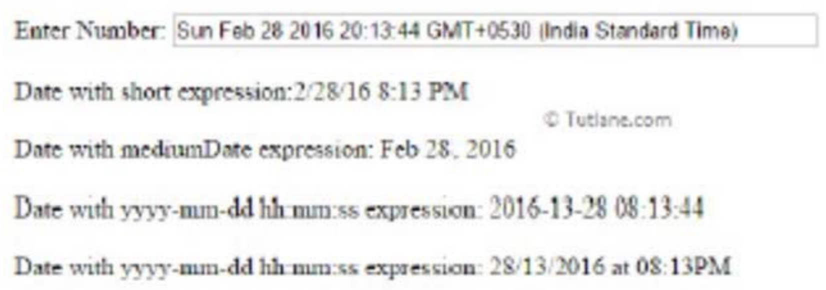
Aim:- Formatting data for display within components and interaction between components.



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>
AngularJs Date filter Example
</title>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<script>
var app = angular.module("AngulardateApp", []);
app.controller("datectrl", function ($scope) {
$scope.sampledate = new Date();
});
</script>
</head>
<body ng-app="AngulardateApp">
<div ng-controller="datectrl">
Enter Number: <input type="text" ng-model="sampledate" style="width:400px" /><br />
<br />
Date with short expression:{{sampledate | date:"short" }}<br /><br />
Date with mediumDate expression: {{sampledate | date : "mediumDate"}} <br /><br />
Date with yyyy-mm-dd hh:mm:ss expression: {{sampledate | date : "yyyy-mm-dd hh:mm:ss"
0}} <br /><br />
Date with yyyy-mm-dd hh:mm:ss expression: {{sampledate | date : "dd/mm/yyyy 'at'
hh:mma" : 0}}
</div>
</body>
</html>
```

Output of AngularJS Date Filter Example

Following is the result of date filter example in angularjs.



```
Enter Number: Sun Feb 28 2016 20:13:44 GMT+0530 (India Standard Time)

Date with short expression:2/28/16 8:13 PM

Date with mediumDate expression: Feb 28, 2016

Date with yyyy-mm-dd hh:mm:ss expression: 2016-13-28 08:13:44

Date with yyyy-mm-dd hh:mm:ss expression: 28/13/2016 at 08:13PM
```

© Tutlane.com

Questions: Viva Voce

Question 1: How can you integrate AngularJS with HTML?

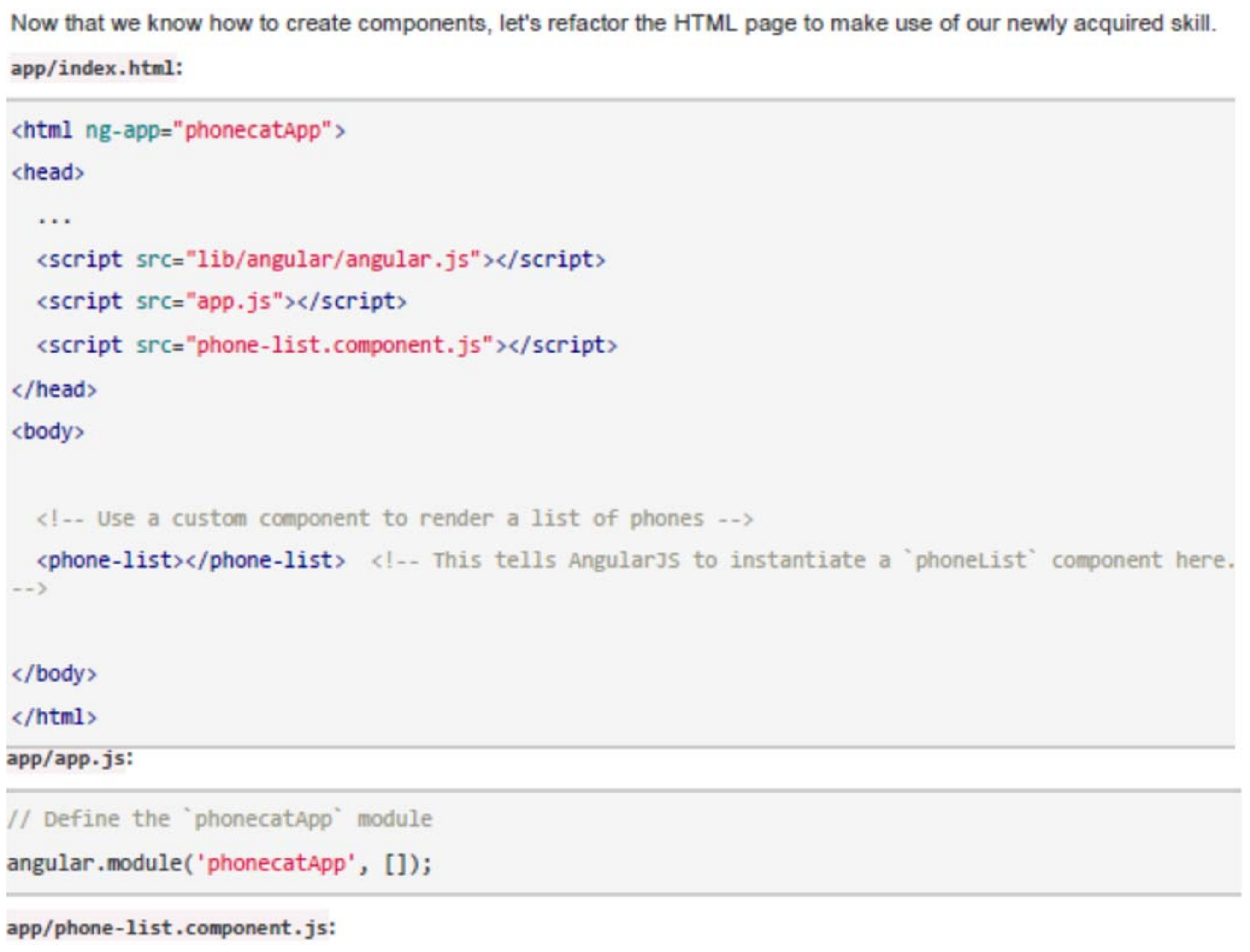
Question 2: What is orderby filter in AngularJS?

EXPERIMENT 3.2

Mapped Course Outcomes-CO3

CO3: The students shall be able to compare within different frameworks available and select best possible framework

Aim:- Reusability of different components within the application.



Now that we know how to create components, let's refactor the HTML page to make use of our newly acquired skill.

`app/index.html`:

```html
<html ng-app="phonecatApp">
<head>
  ...
  <script src="lib/angular/angular.js"></script>
  <script src="app.js"></script>
  <script src="phone-list.component.js"></script>
</head>
<body>


  <!-- Use a custom component to render a list of phones -->
  <phone-list></phone-list>  <!-- This tells AngularJS to instantiate a `phoneList` component here.
-->


</body>
</html>
```

`app/app.js`:

```js
// Define the `phonecatApp` module
angular.module('phonecatApp', []);
```

`app/phone-list.component.js`:

```
// Register `phoneList` component, along with its associated controller and template
angular.
  module('phonecatApp').
  component('phoneList', {  // This name is what AngularJS uses to match to the `<phone-list>`
element.
    template:
        '<ul>' +
          '<li ng-repeat="phone in $ctrl.phones">' +
            '<span>{{phone.name}}</span>' +
            '<p>{{phone.snippet}}</p>' +
          '</li>' +
        '</ul>',
    controller: function PhoneListController() {
      this.phones = [
        {
          name: 'Nexus S',
          snippet: 'Fast just got faster with Nexus S.'
        }, {
          name: 'Motorola XOOM™ with Wi-Fi',
          snippet: 'The Next, Next Generation tablet.'
        }, {
          name: 'MOTOROLA XOOM™',
          snippet: 'The Next, Next Generation tablet.'
        }
      ];
    }
  });
```

Questions: Viva Voce

Question 1: Which platforms are used for large scale cloud computing?

Question 2: Explain System integrators in cloud computing.

EXPERIMENT 3.3

Mapped Course Outcomes-CO3

CO3: The students shall be able to compare within different frameworks available and select best possible framework

Aim:- Responding to an event within the AngularJS framework.

When we want to create advanced AngularJS applications such as User Interaction Forms, then we need to handle DOM events like mouse clicks, moves, keyboard presses, change events and so on. AngularJS has a simple model for how to add event listeners. We can attach an event listener to an HTML element using one of the following AngularJS event listener directives:

ng-click
ng-dbl-click
ng-mousedown
ng-mouseup
ng-mouseenter
ng-mouseleave
ng-mousemove
ng-mouseover
ng-keydown
ng-keyup
ng-keypress
ng-change
Here is a simple AngularJS event listener directive example:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Acgular Event</title>
    <script src="~/Scripts/angular.js"></script>

    <script>
        angular.module("myapp", [])
                .controller("Controller1", function ($scope) {
                    $scope.myData = {};
                    $scope.myData.dvClick = function () {
                        alert("Div clicked");
                }
            });

    </script>
</head>
<body ng-app="myapp">
    <div ng-controller="Controller1">
        <div ng-click="myData.dvClick()">Click here</div>
    </div>
</body>
</html>
```

When we click the text within the div, the myData.dvClick() function will be called. As you can see in the controller function, the myData object has a dvClick() function added to it. The event listener functions called are functions added to the $scope object by the controller function.
The event listener directives have a special variable named $event that we can use as a parameter to the event listener function. The $event variable contains the original browser event object. Here is an example:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Acgular Event</title>
    <script src="~/Scripts/angular.js"></script>

    <script>
        angular.module("myapp", []).controller("Controller1", function ($scope) {
            $scope.myData = {};
            $scope.myData.dvClick = function (event) {
                alert("clicked: " + event.clientX + ", " + event.clientY);
            }
        });
    </script>
</head>
<body ng-app="myapp">
    <div ng-controller="Controller1">
        <div ng-click="myData.dvClick($event)">Click here With Event</div>
    </div>
</body>
</html>
```

We can also other parameters to our event listener functions. Here is an example that adds an event listener function to a list of li elements:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Acgular Event</title>
    <script src="~/Scripts/angular.js"></script>

    <script>
        angular.module("myapp", [])
            .controller("Controller1", function ($scope) {
                $scope.myData = {};
                $scope.myData.items = [{ CName: "Smith" }, { CName: "John" }, { CName: "Tony" }];

                $scope.myData.dvClick = function (item, event) {
                    alert("clicked on : " + item.CName + " + " + event.clientX + ": " + event.clientY);
                }
            });
    </script>
</head>
<body ng-app="myapp">
    <div ng-controller="Controller1">
        <ul>
            <li ng-repeat="item in myData.items"
                ng-click="myData.dvClick(item, $event)">{{item.CName}}</li>
        </ul>
    </div>
</body>
</html>
```

Questions: Viva Voce

Question 1: Why API's is used in cloud services?

Question 2: What are the advantages of cloud services?

Further Reading:

1. Online Tutorials Library (tutorialspoint.com)
2. Iterate or Loop through each Array item or Object using AngularJS forEach() (encodedna.com)
3. Angular - Introduction to Angular concepts