**Answers for Questions**

1. Angular is a JavaScript framework which allows you to create reactive single-page-applications. It allows us to display dynamic data using typescript file
2. We can navigate around to multiple modules/functionalities in a single page html file. URL changes but it is same page(one single html). It gives the user very reactive experience. Very fast. Java scripts changes DOM, DOM changes which changes html in browser.
3. Started with Angular JS (Angular 1) has fundamental flaws. Angular 2 (re-write). Angular 4 …10,11,12…
4. It is used to bundle up the angular cli.5) It has npm(node package manager) which gives dependencies
5. Node js has npm(node package manager) which gives dependencies
6. Ng new project-name
7. Ng serve
8. In package.json
9. Dependencies for which angular uses
10. For end-to-end testing
11. Index.html
12. We use selector(app-root) adding inside index.html
13. Using {{string}}
14. It is a directive given by angular. It gives us bidirectional data binding. In input text if we give ngModel=”name” , it will update name in ts file and update html based on ts file
15. FormsModule from angular/forms
16. It is a super set of java script. Types, classes, Interfaces. It doesn’t run in browser. It compiled to javascript through cli. Java script is run in browser.
17. Bootstrap is used for better styling. Cmd to install: npm install --save bootstrap@3 . configure it to angular.json in styles block. To check if bootstrap is applied, inspect-sources-style.css-u can see bootstrap in comments.
18. Server initiates index.html where in the body we have app-root which is a selector of app.component. If u view the page-source, u can see scripts imported. They r imported by cli as bundles when we use serve cmd. These script imports have our code. First code executed is main.ts code.
19. Angular gets started then first code file it reads is main.ts file, in this file we mentioned AppModule which refers to App component. So App component rendered.
20. Components are key features. App component is root component. These allows us to separate complex application into diff parts and reusable.
21. Ng generate component server (or) ng g c server . ts file, html,css..
22. @Angular/core. Angular/core gves us core functionalities
23. Selector is unique-name we give to a component. Diff ways to write selector-check in servers component. and templateURL is html file path
24. Module is a bundle of functionalities. It says which features my Angular application uses. Angular bundles modules and have them as packages. We declare components in NgModule which comes from @angular/core.
25. Just creating a component, Angular will not load it. So we need to add the component in NgModule declarations
26. No webpack automatically takes the extensions while packing the bundle of code
27. It lets us to add other modules to our Application
28. If selector name is app-server then <app-server></app-server>
29. Using template instead of templateUrl u can directly write html code, Using ‘ single line html code ’. To use multiple lines of html code use ` code here `
30. Using styleUrls to give style.css file path. To use styling directly inside ts file use styles instead of styleUrls. styles: [` code here `]
31. Communication between typescript code and html template
32. One way and two-way data binding
33. String interpolation is a one way data binding technique to output the data from type script file to html code (use of {{serverID}} in html ) , u can also use method which returns string {{ getServerStatus() }
34. <button class="btn btn-primary" disabled>Add Server</button>
35. <button class="btn btn-primary" [disabled]="!allowNewServer">Add Server</button>
36. String interpolation: <p>{{allowNewServer}}</p>

Property binding : <p [innerHTML]="allowNewServer"></p>

If you want to change attribute vale then use property binding

1. For any events triggered by user we want something to happen

<button class="btn btn-primary" (click)="onCreateServer()">Add Server</button>

In ts file

onCreateServer() {

    this.serverCreationStatus = 'Server was created!';

  }

one-way data binding (this only updates the serverName value)

<input

type="text"

class="form-control"

(input)="onUpdateServerName($event)">

<p>{{serverName}}</p>

two-way data binding(this updates and displays the serverName value at same time)

<input

type="text"

class="form-control"

[(ngModel)]="serverName">

1. Directives are instructions in the DOM. Components(Directive with template) are instructions in the DOM.
2. Because it is a structural directive. It changes the structure of DOM

<p \*ngIf="serverCreated">Server was created, server name is {{serverName}}</p>

Ng-template is used to mark div in html

<p \*ngIf="serverCreated; else noServer">Server was created, server name is {{serverName}}</p>

<ng-template #noServer>

    <p #noServer>No Server was created</p>

</ng-template>

1. ngStyle is used to add styling values dynamically through ts file

<p [ngStyle]="{backgroundColor: getColor()}">Server with Id {{ serverId }} is {{ getServerStatus() }}</p>

1. Below online class gets added based on serverStatus condition

<p [ngClass]="{online: serverStatus === 'online'}">Server with Id {{ serverId }} is {{ getServerStatus() }}</p>

1. <app-server \*ngFor="let server of servers"></app-server>

<div \*ngFor="let logItem  of log; let i = index"

[ngStyle]="{backgroundColor: i>=4 ? 'blue' : 'transparent'}"

[ngClass]="{'whiteText': i>=4}"

>{{logItem}}</div>

1. ng g c recipes --skip-tests
2. ng g c recipes/recipe-list --skip-tests
3. It has 12 columns. col-md is for medium screen responsiveness. Md-5 is 5 columns and md-7 is seven columns width

<div class="row">

    <div class="col-md-5">

        <app-recipe-list></app-recipe-list>

    </div>

    <div class="col-md-7">

        <app-recipe-item></app-recipe-item>

    </div>

  </div>

1. <hr>
2. Model is a structure/ Object we use throughout the application. Blueprint of an object

export class Recipe {

    public name : string;

    public description: string;

    public imagePath : string;

    constructor(name: string, desc: string, imagePath: string){

        this.name = name;

        this.description = desc;

        this.imagePath = imagePath;

    }

}

1. <a

        href="#"

        class="list-group-item clearfix"

        \*ngFor="let recipe of recipes">

        <div class="pull-left">

          <h4 class="list-group-item-heading">{{ recipe.name }}</h4>

          <p class="list-group-item-text">{{ recipe.description }}</p>

        </div>

1. <a

        href="#"

        class="list-group-item clearfix"

        \*ngFor="let recipe of recipes">

        <span class="pull-right">

          <img

            [src]="recipe.imagePath"

            alt="{{ recipe.name }}"

            class="img-responsive"

            style="max-height: 50px;">

        </span>

      </a>