DOMS | Page No.
Date / /

Name :- Khokhariya Vishwa B.

Sem :- 7th Roll no :- 23

Sub :- 701 Date :- 17-10-23

* write in details with diagrams, codes and example wherever applicable:

① Angular introduction.

- Angular is a popular open-source web application framework for building dynamic, single-page application and enterprise-grade web application.

- It is developed and maintained by google and a community of individual developers and organization.

- Angular is written in TYPESCRIPT, which is a superset of JAVASCRIPT and provides a comprehensive set of tools and features for developing complex and feature-rich web application.

Key Features of Angular.

- Angular offers a wide range of feature-rich web application.

① Declarative UI:-

- Angular allows developers to build the user interfaces using HTML templates and declarative data bindings.
- This simplifies the process of creating dynamic and responsive user interfaces.

② Component-Based Architecture.

- Angular applications are build using a components-based architecture.
- components are self-contained, reusable building blocks that encapsulate HTML Template, styles and behaviour.

③ Dependency Injection:-

- Angular's built in dependency injection system makes it easy to manage and share application-wide services and data between different of the application.

023_vishnu_Khokhariya

④ Two-way Data Binding

- Angular supports two-way data binding, allowing automatic synchronization of data between the model and the view.
- This simplifies the process of keeping the UI in sync with the underlying data.

⑤ Routing

- Angular provides a powerful routing system that enables the creation of multipage application with support for a linking, loading and route guards.

⑥ Forms

- Angular offers robust support for building complex forms, including form validation, control, data binding.

⑦ HTTP client

- Angular includes an HTTP client module for making HTTP request.

⑧ Testing

- Angular has built-in support for unit testing and end-to-end testing, making.

DOMS Page No.
Date / /

(2) Angular Application structure

- Angular application follow a well-defined structure that helps organize your code and resources effiecently.

- This structure ensure a clear separation of concerns, making it easier to maintain, test and scale your application.

```
angular-app/
    |-src/
      |-app/
        |-components/
          |-my-components/
            |-my-component.component.ts
            |-my-component.component.html
            |-my-component.component.css
            |-my-component.component.spec.ts
          |-services/
          |-modules/
        |-assets/
        |-environments/
        |-index.html/
        |-main.ts
    |-angular.json
    |-package.json
    |-tsconfig.json
    |-tsconfig.app.json
    |-tslint.json
    |-README.md
```

- my-component.component.css : css/scss file for styling component.

- my-component.component.spec.ts : unit test for component.

④ 'services' directory

- The services directory is where you define Feature Angular services.
- Services are used for handling data, making HTTP requests and sharing Functionality across components.

⑤ modules directory

- The module directory is where you define Feature modules.
- Feature modules help you organize and encapsulate related components, services and routes help you
- Each Feature module has its own directory

⑥ assets directory.

- The assets directory is used to store Static files, such as images, fonts or configuration files.

DOMS Page No.
Date / /

① 'src' Directory

- The src directory is the heart of Angular application.
- It contains all the source code and resources required for application.

② 'app' Directory.

- The app directory is where will spend most of time.
- It contain the core components, services and modules of application.

- Component Directory.

- The component directory is where you define your angular components.
- Each component is organized within its own subdirectory.
- For example, if you have component named 'my-component'. you will have the following files:

my-component.component.ts : TYPEscript file that defines component class

- my-component.component.html : HTML template associated.

023. Vishwa Khokhariya

③ environment Directory.

- The environment directory is containing environment specific configuration files.
- By default, Angular provides two files: environment.ts for developing and environment.prod.ts for application production

④ index.html and main.ts

index.html

- The main HTML File for your application
- it serves as the entry point for your Angular application.
- you can include metadata, css files, and the angular application root component selector.

main.ts

- The file is the entry point of your application Typescript code.
- it bootstraps the angular application and starts the execution.

④ Configuration files.

- angular.json :- This file contains

configuration setting for your Angular Project, including build options, assets and dependencies.

- Package.json:- This is your project's package.json file. which lists your projects dependencies and scripts.

- tsconfig.json: The Typescript configuration file for your entire project.

- tsconfig.app.json: and tsconfig.spec.json: specific Typescript configuration for the app and tests respectively.

- tslint.json: configuration for TSLint, a code analysis tool for typescript.

5) README.md

- The readme.md file is used to provide documentation and instruction for your project.

023-vishnu khokhariya

(3) Angular Architecture:

- Angular architecture follows the MVC pattern; but in Angular terminology, its refered to as model-view-viewmodel

1. Components: A component encapsulates the user interface, its behavior and the data it display.
   - Each component consist of a TypeScript class, HTML page, and associated style.

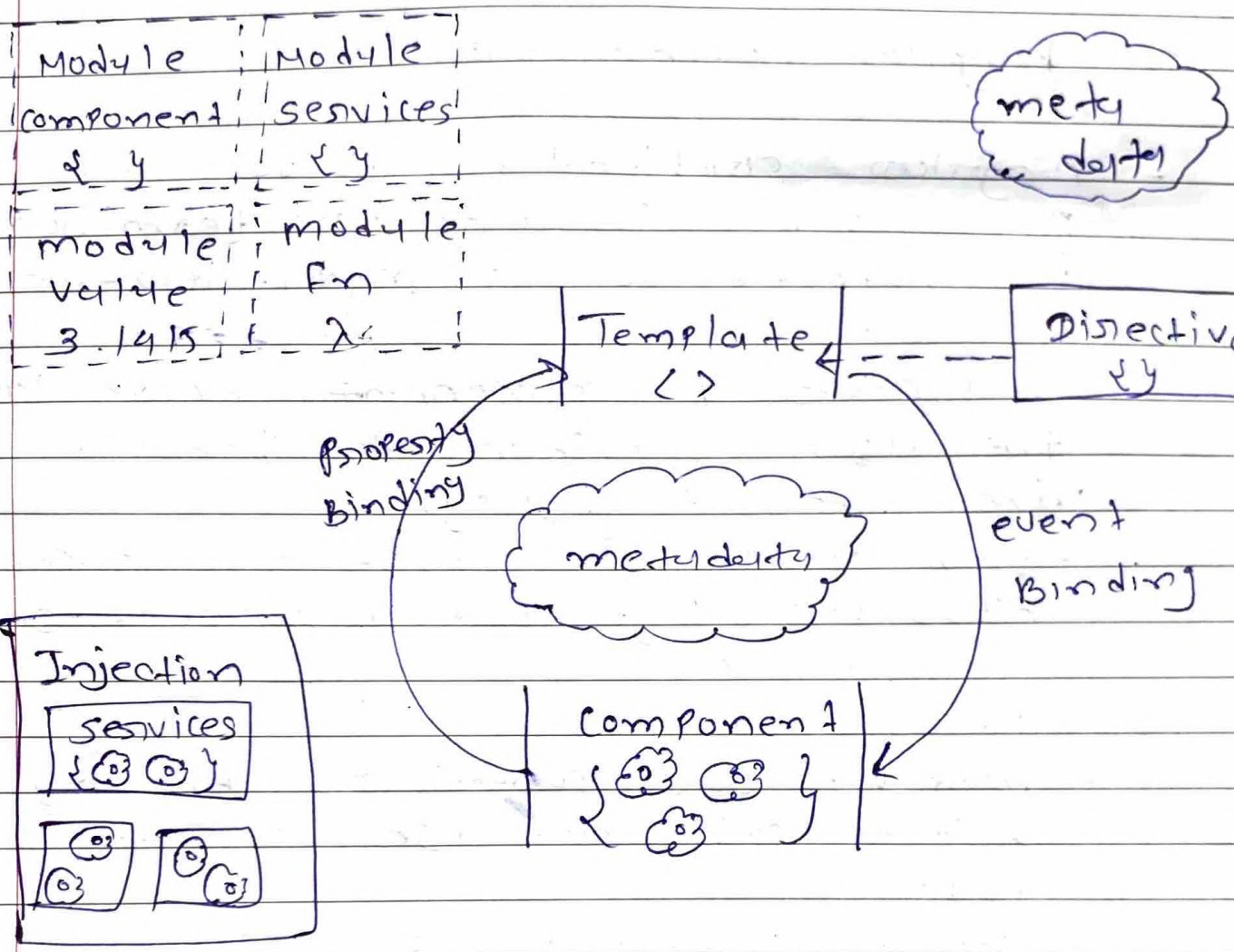
2. Modules:-
   - modules are containers for organizing related component, services and other code
   - The root module is 'APP module'.

3. services:
   - services are used for encapsulating business logic, data access and communication btw component.

4. Directives:
   - Directives are used to manipulate DOM.
   - Angular provides built-in services like ngIF and ngfor.

023_Vishwa Khokhariya

DOMS Page No.
Date / /



- Component :-

Class : contains Properties and methods to define the component's behaviour

Template: Angular specific syntax to bind data and control the view

Style : css / scss styles for HTML Template

Metadata : used to decorate the class and define the components characteristics.

023-vishwa Khokhariya

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

@Component({
  selector : 'app-greeting',
  template : '<h1> {{message}} </h1>',
  style : ['h1 { color : blue; }']
})

export class Greeting Component{
    message : "Hello, Angular';
}
```

- Module example

```
import {NgModule} from '@angular/core';
import { BrowserModule} from '@ angular/
                              Platform-browser';
import { Greeting Component} from
                    './greeting.component';

@NgModule ({

    declaration : [Greeting Component],
    imports : [Browser module],
    bootstrap : [Greeting Component]

})

exports class APPmodule { }
```

DOMS Page No.
Date / /

- service example

```
import { Injectable } from '@angular/core';

@Injectable()

export class DataServices {

    Private data: string[] =
        ['Item1', 'Item2', 'Item3'];

getData() {
    return this.data;
}

addData(newItem : string)
{
    this.data.Push(newItem);
}
}
```

Directives example

```
<ul>
  <li myFor = "let item of items">
  {{item}}
<li>
</ul>
```

023-Vishwa Khokhariya

- Templates example

```
<div>
    <h1> {{ title.y }} </h1>
    <p> {{ description }} </p>
</div>
```

- metadata example

```
import {component} from '@angular/core';

@component ({

  selector : 'app-greeting'
  Template : '<h1> {{ message } } </h1>'
  style    : ['h1 { color : blue; }']

})

export class greetingcomponent
{
    message : 'Hellow, Angular';
}
```

(4) Typescript introduction

- Typescript is an open-source, statically typed superset of javascript that compiles to plain JS.
- It was developed by microsoft and is designed for building large-scale, maintainable and robust web application.
- Typescript add static typing interfaces, classes and other feature to JS, making it a powerful tool for developers.

* Key features of Typescript

- Static typing, Interfaces, classes. Type Annotation, modules, Enems, Type Interface, Compatibility.

Typescript	V/S	Java script
- statically typed		- dynamically typed
- supports OOP		- doesn't support
- must be transpiled to JS before execution		- executed directly.

023-Vishal Khokhariya

Example

Type Annotation

```
let name : string = 'John';
let age : number = 50;
let is student : boolean : false;

Function greet (Person : string ):
        string {

    return 'Hello, ${ Person }!';
}
console.log (greet (name));
```

- Interfaces

```
interface Person {

    name : string;
    age : number;

Function introduce (Person : Person ):
    string {

    return 'Hi, Im ${ Person.name }';
}

const user : Person : {name : 'Alice', age :25}
console.log ( introduce (user));
```

DOMS Page No.
Date / /

classes

```
class Animal
{
  constructor (public name : string ){ }

  makeSound ( sound : string ) : void
  {
    console.log (' ${this.name}
          make ${sound} );
  }
}

const cat : new Animal ('cat');
cat.makeSound ('meow');
```

Typescript workflow

```
npm install -g typescript
write code
  compile → tsc file.ts
  Run.
```

023 - Vishwa Khokhariya

(5) Angular CLI commands

- The angular CLI is a powerful tool that simplifies the development, testing and deployment of angular application

- installation

npm install -g @ angular / cli

- new project creation

ng new my -app

- change the Project directory

cd my-app

- Development commands

serve : starts development server and serves

ng serve --open

generate : used to create various component

ng generate component my-component

DOMS Page No.
Date / /

- Testing commands

  Unit tests

  ng test

- End-to-End Tests

  ng e2e

- Building & deployment

  ng build

  Deploy:

  The deployment process typically involved copying the contents of the disk directory generated by the 'ng build'

- code linking & formatting

  ng Lint

  ng format.

Page No.
Date

023-Vibhor Khokhariya

⑥ Component Lifecycle Hooks

- In Angular, components have a lifecycle that you can tap into by implementing certain interface and methods.

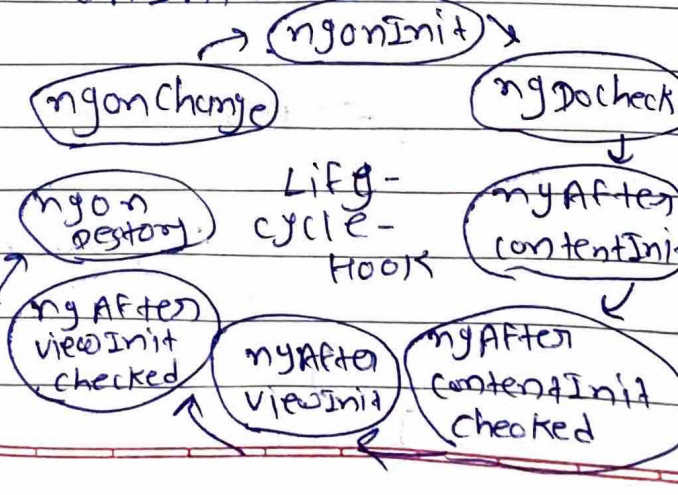
Initialization.

① constructor()
- similar to React, constructor is where you initialize your component.

```
export class mycomponent implement
        onInit { constructor() { }
               }
```

② ngOnInit()
- The 'ngOnInit' method is part of the onInit lifecycle hook and is called after the constructor. It's commonly used for initializing data

```
export class mycomponent implements
                                onInit
{
   ngOnInit()
   {
   }
}
```



Page No.
Date

★ onchanges

① ngonchanges()
- This is called when any data-bound property of the component changes. It receive a 'simpleChange' object containing previous and currently property value.

```
export class mycomponent implements
                onchange {
  ngonchange (changes : simplechanges)
  {
  }
}
```

★ Docheck

① ngDocheck()
- The 'ngDocheck' lifecycle hook is triggered during every change detection cycle. It allows you to implement your own change detection logic

```
export class myComponent implements
                Docheck {
    ngDocheck()
  {
  }
}
```

✱ After ViewInit and viewchecked.

① ngAfterViewInit()
- invoked after the component's view has been initialized

```
export class A implements AfterView {
    ngAfterViewInit() {
    }
}
```

② ngAfterviewchecked()
- invoked after the view has been checked by Angular

```
export class A implements AfterviewChecke
    ngAfterviewchecked() {
    }
}
```

✱ Destruction

① ngonDestroy()
- called just before the component is destroyed. It's used for cleanup tasks like unsubscribing from observables

```
export class mycomponent implements
                                onDestroy {
    ngonDestroy() {
    }
}
```

Write details of following with example commands, flow diagram, code fragments and at the end make application combining them.

① Components, Templates

* Components:

① Create a new Angular application

```
ng new my-angular-app
cd my-angular-app
```

② Generate a new component

```
ng generate component my-component
```

③ Component structure :-

```
import { Component, onInit } from '@
                    angular/core';
@Component ( {
  selector : 'app-my-component',
Templat(URl : './my-component.component.
                    html',
styleUrls : ['./my-component.component.css']
  } )
Export class mycomponent implements onInit {
  constructor ( ) { }
  ng onInit ( ) { }
}
```

Templates:-

① Edit the component's template

```
<div>
<h2> Welcome to Angular ! </h2>
<P> This is my custom component. </P>
</div>
```

② Using Angular template Syntax:

```
<div>
<h2> {{title}} </h2>
<P> {{ description }} </P>
</div>
```

③ component Data Binding

```
export class mycomponent implements
                                OnInit {
  title : String = "welcome to Angular !";
description : String = 'This is my
                        custom component.';

constructor () {
                }
myonInit () {
            }
    }
```

Flow Diagram

- combining components and Templates:

① include the component in the main app component

```
<div>
<app-my-component></app-my-component>
</div>
```

② Update the app component class.

```
import {component} from '@angular/core';
@component({
  selector: 'app-root',
  templateURL: './app.component.html',
  styleURLS: ['./app.component.css']
})

export class AppComponent{
   title = 'myAngularAPP';
}
```

③ Run the application

```
ng serve
```

(2) Data Binding, Event Binding, expression, interpolation.

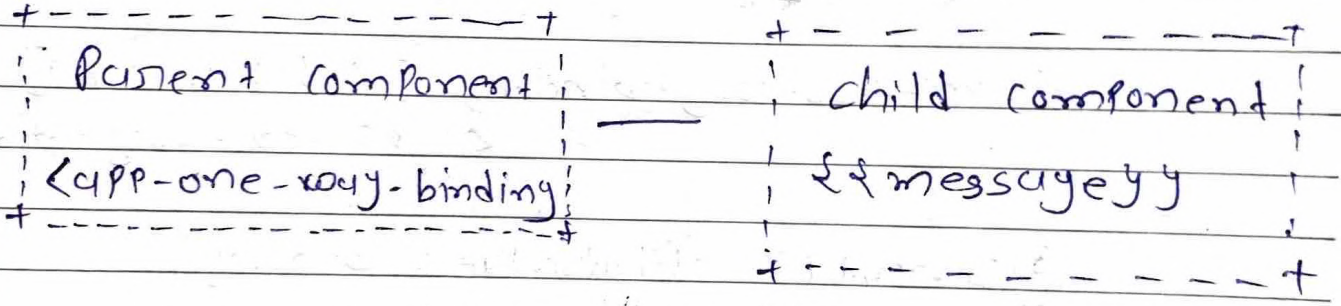
* Data Binding:

(1) One-way Binding:

- Example command:

ng generate component one-way-binding

- Flow Diagram:



- code Fragment:

```
<app-one-way-binding [message]="parentmsg">
</app-one-way-binding>
```

```
@Input() message : string;
```

(2) Event Binding:

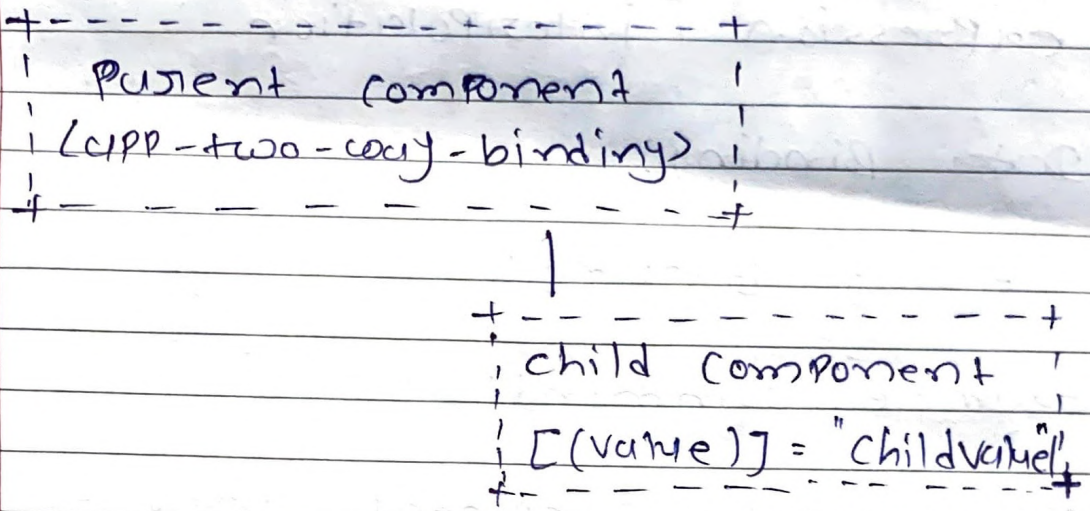
(1) Two-way Binding:

- Example command:

ng generate component two-way-binding

Page No.
Date

- Flow Diagram



- Code fragment:

```
<app-two-way-binding [(childvalue)] = "Pvalue">
                    </app-two-way-binding>

@Input() childvalue : string;
@Output() childvaluechange = new
                    EventEmitter<string>();
```

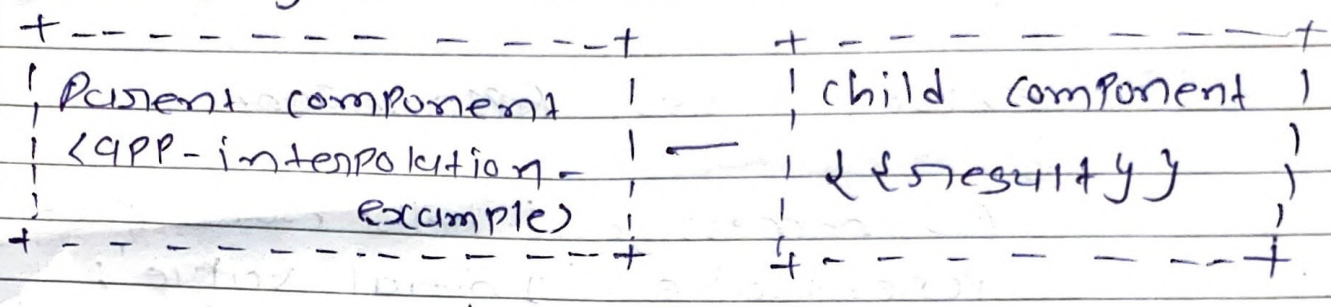
③ Expression and Interpolation:

1. interpolation and expressions:

- Example command:

ng generate component interpolation-example

- Flow Diagram



- Code Fragment:

```
<app-interpolation-example [input] = "pInput">
                    </app-interpolation-example>

@Input() input : number;
  get result() : number {
        return this.input * 2;
    }
```

④ Combing components.

1. include components in the main App component

```
<div>
<app-one-way-binding [message] = "pmessage">
        </app-one-way-binding>
<app-two-way-binding [(childvalue)] = "pvalue">
                    </app-two-way-binding>
<app-interpolation-example [input] = "pInput">
                    </app-interpolation-example>
</div>
```

2. Update the APP component class

```
export class AppComponent
{
  ParentMessage = 'Hello From Parent';
  ParentValue = 'Initial value';
  ParentInput = 10;
}
```

3. Run the Application:

```
ng serve
```

(4) Routing

(1) Create a new Angular application

```
ng new my-routing-app
cd my-routing-app
```

(2) Generate components for Routing

```
ng generate component home
ng generate component about
ng generate component contact
```

(3) Setup Routing:

- Open 'app.module.ts' and import

'RouterModule' and 'Route' From '@angular/router':

import {RouterModule, Routes} from '@angular/router';

const routes: Routes = [
{path: 'home', component: HomComponent},
{path: 'about', component: About},
{path: 'contact', component: Contact}];

- Use Routing outlet

<h1> welcome </h1>
<router-outlet> </router-outlet>

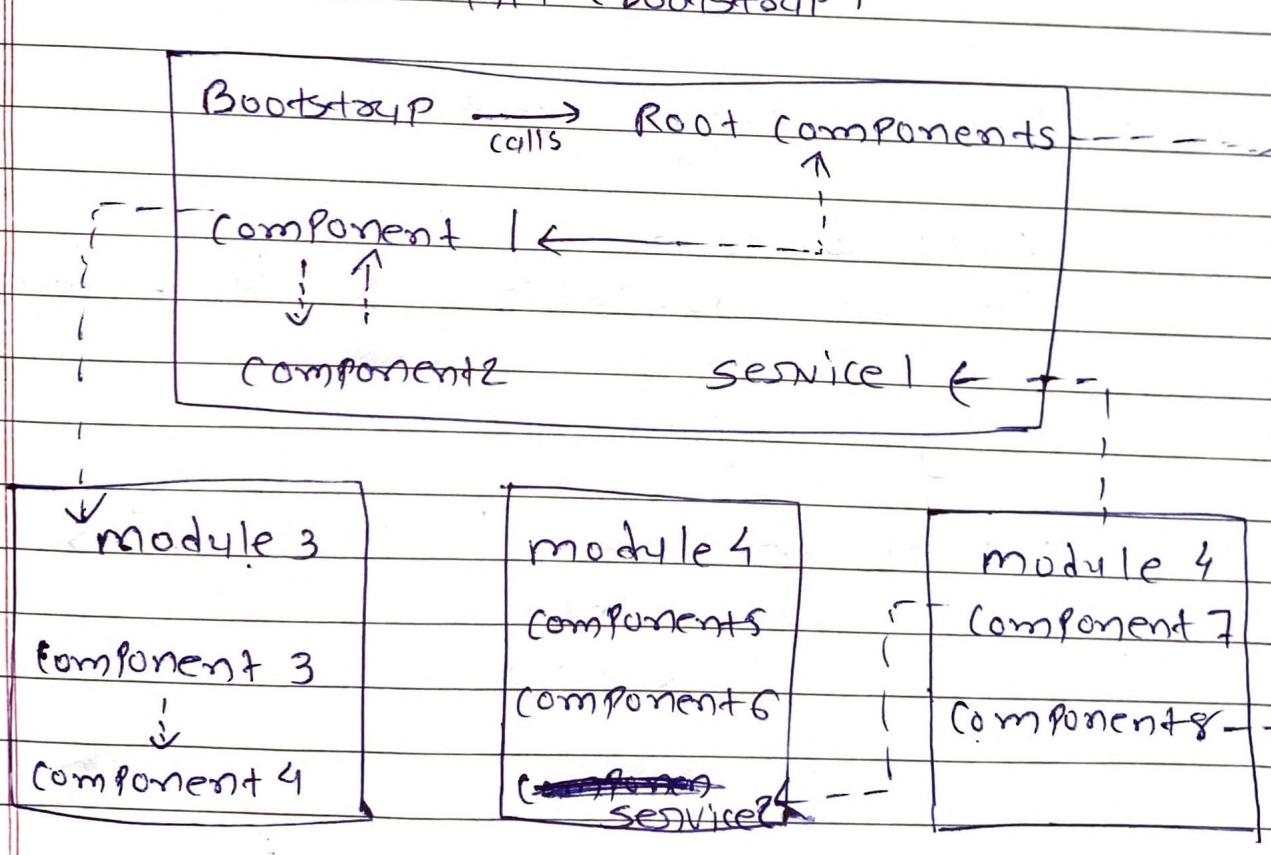
- Navigation

<a routerLink = "/Product"> to

Page No.
Date

(4) Modules :

- Angular Module is basically a collection of related feature / Functionality.
- Angular module groups multiple components and services under a single context.



Page No.
Date

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

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

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

@NgModule({

    declarations: [
        APPComponent
    ],
    imports: [
        BrowserModule
    ],
    Providers: [],
    bootstraps: [APPComponent]
})

export class APPModule { }
```

⑥ Directive

- Angular directives begin with ng- where ng stands for Angular and extends HTML attributes with @ directive decorator.

- directive enables logic to be included in the Angular templates.

- Three categories

- attribute directive

- used to add new attributes for the existing HTML element to change its look and behavior.

```
<HTMLTag [alterDirective] = 'value' />
```

- Structural Directive

```
<HTMLTag [structuredDirective] = 'value' />
```

component - based - Directive

```
<component-selector-name [input-reference
    = "input-value"> ---- </component>
```

⑦ Forms

- Forms are used to handle user input data
- Two types of forms:-

Template driven Forms

- It is created using directive in the template.
- mainly used for creating a simple form application.

Reactive Forms

- Reactive forms are created inside a component class so it is also refered as model driven forms.
- Every control will have an object in the component and this provides greater control and flexibility in the form programming

9. HttpClient

- Powerful and convenient way to make HTTP requests in your Angular application.
- simplifies the process of sending and receiving data from a server, whether it's fetching data from an API, Posting data, or handling other HTTP request.

Installation

npm install @angular/common/http

importing

import { HttpClientModule } from @angular/common/http

Use

constructor (private http: HttpClient.) {}

- combine all above code fragements in example. it will make an Angular app.

⑤ Pipes

- Pipes are referred as Filters. It helps to transform data and manage data within interpolation, denoted {{ I }}

- It accepts data array, integers and strings as input which are separated by 'I' symbol.

Example

```
export class TestComponent {

    PresentDate = new Date();
}

<div>
  Today's date :- {{ PresentDate }}
</div>
```

adding pipe Parameter

```
<div>
  Today's date :- {{ PresentDate | date }}
</div>
```