01)
02)

Introduction of Angular

PAGE No.
DATE / /

* ⇒ Angular is an open source JavaScript MVC framework web application. It extends the HTML and makes it dynamic Angular is used to create Single page Application (SPA).

⇒ build a complex web and mobile application (UI)

* ⇒ Angular is a JavaScript framework written in Typescript and maintained by Google.

⇒ Using Angular you can build client-side applications using HTML, CSS, and Typescript.

⇒ **Features of Angular**

1. Document Object Model (DOM)
2. TypeScript
3. Data Binding
4. Testing
5. MVC
6. High Speed, Ultimate Performance.
7. Universal Support.
8. Cross Platform (Web, Mobile & Desktop)

⇒ **Building Blocks of Angular App**

1) Modules
2) Components
3) Templates
4) Metadata
5) Services
6) Dependency injection
7) Directives

PAGE No.
DATE / /

⇒ Apps Made with Angular

- Gmail App
- Weather.com
- Paypal
- Forbes
- MS office
- Netfix
- ⋮

⇒ Angular Versioning (Major, Minor & Patch)

14 . 1 . 0

(Major) (minor) (patch)

(Major)	(minor)	(patch)
- significant New feature	- Smaller features with backward compatibility.	- Bug fixes or any defect will be removed from the the current version.
- Some but Minimal Devolope assistance needed.	- No developer Assistance required	
- You may need to run Upadage script, refactoring code additional Tests & newly added APIs	- No need to run update script	

03) Angular Installation

PAGE No.
DATE / /

→ Code Editor (Vs code)
→ Node JS (☑ Automatically install)
→ NPM (Node Package Manager)
→ Angular CLI (Command Line interface)

⇒ After Installing Node JS & NPM, check their Version
- node -v
- npm -v

⇒ Install Angular CLI
→ create folder (Angular - Projects)

→ Open terminal and run following command: (cmd)
• npm install -g @angular/cli

→ Use the command "ng v" to check the version of angular CLI.

04) Creating first Angular App

PAGE No.
DATE / /

⇒ To create a new workspace and initial starter app:

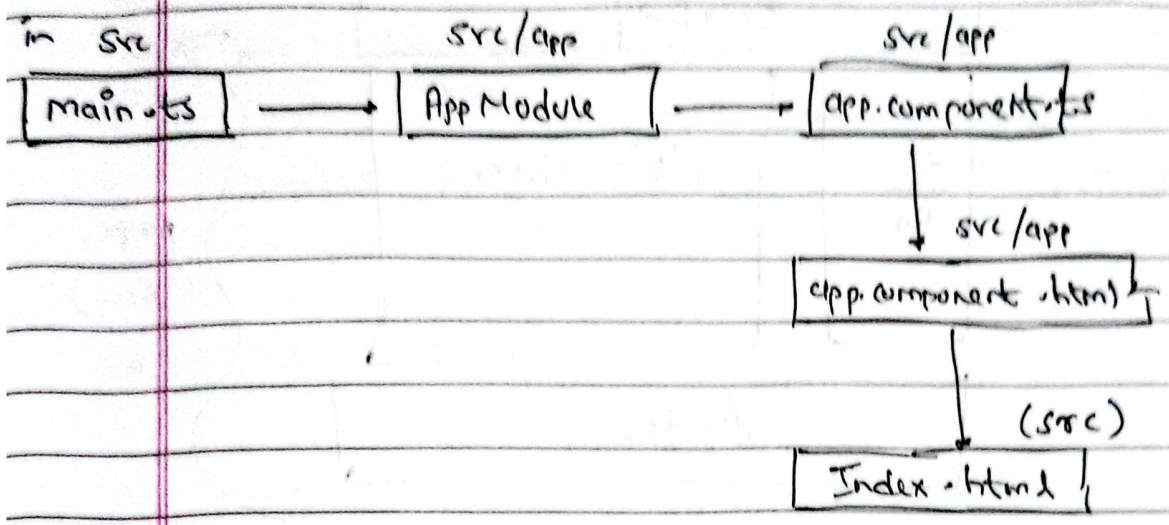
- Run the CLI command
  - ng new new-app

⇒ The angular CLI includes a server, for you to build and serve your app locally.

⇒ Navigate to the ~~f~~ workspace folder (new-app)

- Run the following command
  - ng serve → (launches server, watches your files, rebuilds the app)

05) Workflow of angular app



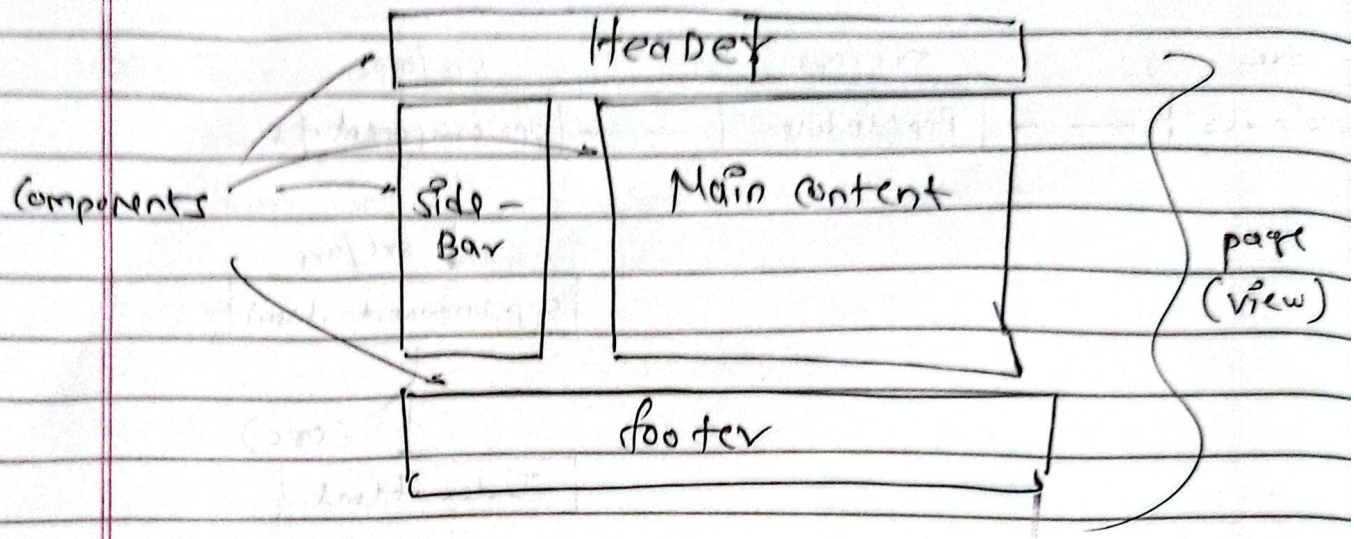
⇒ Every Angular App has 1 module file called AppModule/ root module

⇒ Every Angular App has 1 pre-defined component called app/ ~~or~~ root component.

⇒ Modules can have one or more components and services

⇒ Components have HTML and class.

06) Angular Component.



- view made of one or more components.
- An Angular component represent a portion of a view.
- The component is the basic building block of Angular
- Angular Component = HTML Template + Component class + Component Metadata (Decorator)
- component can be re-used.

⟹ **Generate Angular Component using Angular CLI**

- ng generate component <Component name>
- ng g c <component name>

PAGE No.
DATE / /

⇒ **Html Template** :- html template is nothing but a regular html code with additional Angular specific syntax to communicate with the component class.

⇒ **Class** :- a components class is a TypeScript class that includes properties and method. Properties store data and methods include the logic for the component. Eventually, this class will be compiled into JavaScript.

⇒ **Metadata** :- Metadata is some extra data for a component used by Angular to execute the component, such as the location of Html and css file, selector, provider, etc.

→ Metadata information provided by component decorator function.

* ⇒ There are 3 ways of calling our components inside app.component.html.

① By using custom tag (<app-user> </app-user>)

② By using class selector

selector : .app-user

← . . .

③ By using Attribute :

selector : '[app-user]'

<div app-user> <\div>

07) Inline Template & Inline style

PAGE No.
DATE / /

⇒ in some cases, we don't need a separate file of HTML or CSS in our angular component, we want to implement Html or CSS directly in our "...component" file

⇒ There ~~as~~ are 3 ways of implementing inline template or inline style.

① we need separate HTML file but no need of separate CSS file

```
Styles: [`
        p {
          color: red;
          font-size: 40px;
        }
      `]
```

② we need separate CSS file but no need of separate HTML file

```
template: `<h2> Hello World </h2>`,
```

③ we don't need both HTML and CSS files.

```
template: `h2 class="title"> Helloworld</h2>
Styles: [`
        .title { : color ...
               } : font..
        `]
```

Note :-

① if you don't want CSS file in component (ng g c user1 --inline-style)

② don't need html file (ng g c user2 --inline-template)

③ don't need both

ng g c user --inline-template --inline-style

08) Interpolation

PAGE No.
DATE / /

⇒ Create interpolation component [Display property's value]

→ in .ts class →

```
public name = 'this is interpolation' ; age = 44 ;
```

→ in .html

```
<h2> {{ name }} </h2>
```

⇒ we can write expression

```
<h2> {{ 2+2 / 4 * 5 }} </h2>
```

⇒ Concatenation

```
<h2> {{ name + " And " + age }} </h2>
```

⇒ we can use Javascript properties and methods.

```
<h2> {{ name.length }} </h2>
<h2> {{ name.toUpperCase() }} </h2>
```

⇒ we can use custom method.

→ in class

```
helloMessage () {
    return " Hellow " ;
}
```

→ .html

```
<h2> {{ hellowMessage () }} </h2>
```

⇒ **Interpolation / string interporation in Angular**

→ interpolation allows us to include expressions as part of any string literal, which we use in our HTML.

PAGE No.
DATE / /

→ It's evaluates the expressions into a string

* → Interpolation is one-way binding

→ Interpolation is one way as values go from the component to the template. when the component values change, the Angular updates the view. But if the values changes in the view components are not updated.

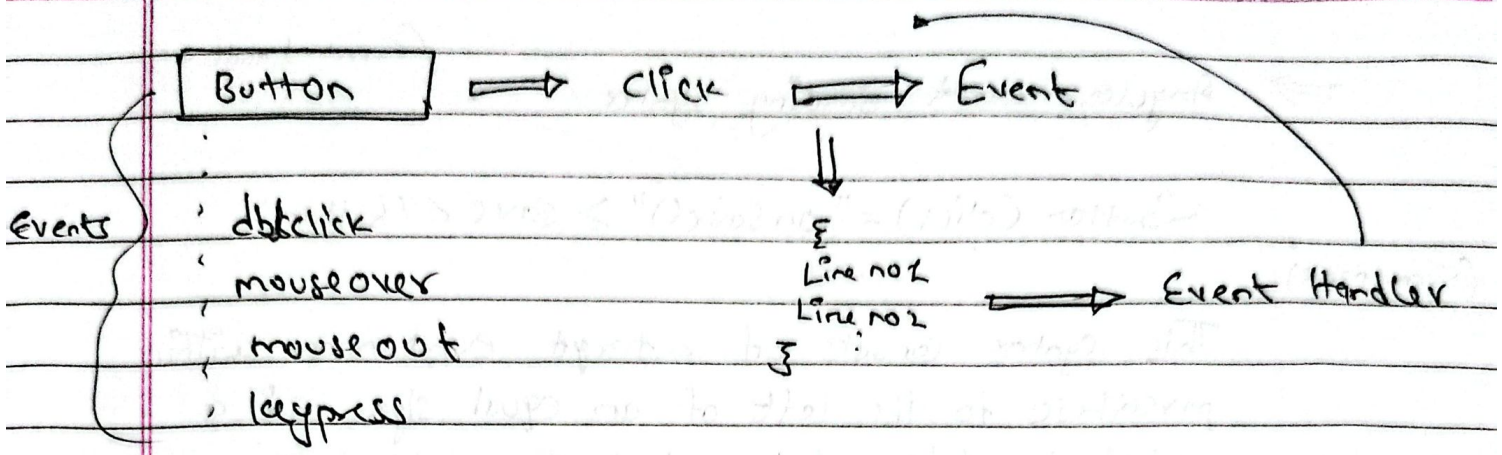
* ⟹ ~~to~~ Cannot use of the following

→ Assignments (=, +=, -=, ...)

→ keywords like new, typeof, instanceof, etc
→ chaining expressions with ";" or ","
→ increment, decrement (++, --)
→ bitwise operators such as | and &

→ Cannot sent objects

29) Event Handling | Event Binding

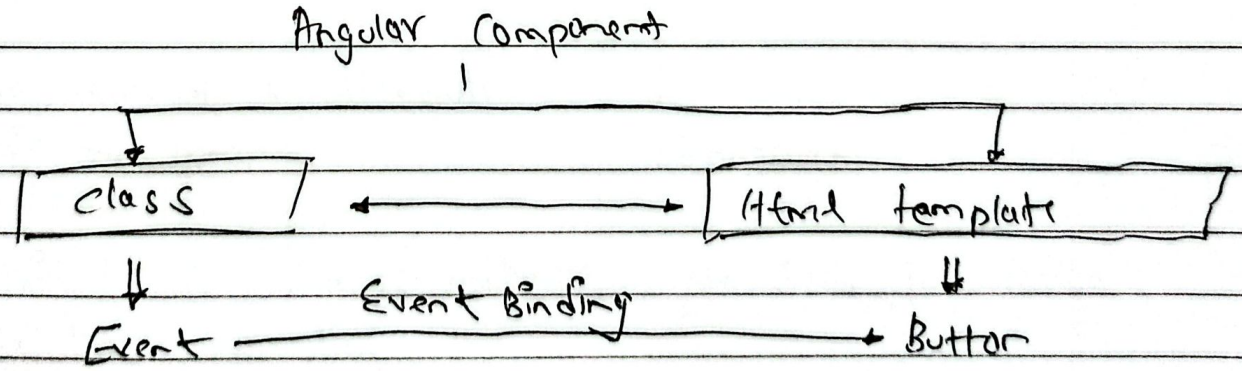


⇒ An event is an action or event detected by a program

⇒ Events can be user actions, such as as clicking a mouse button or pressing a key, or they can be events on the system, such as running out of memory

⇒ <u>Event Binding</u>

→ Event binding lets you listen for and respond to user actions such as keystrokes, mouse movements, clicks, and touches.



PAGE No.
DATE / /

⇒ Angular event binding syntax

(Event handler)

<button (click) = "onSave()" > Save </button>

(Event name)

This syntax consists of a target event name within parentheses to the left of an equal sign and a quoted template statement to the right.

10) Property Binding

PAGE No.
DATE / /

<input type="text" id="exampleID">

Html Attribute — DOM property

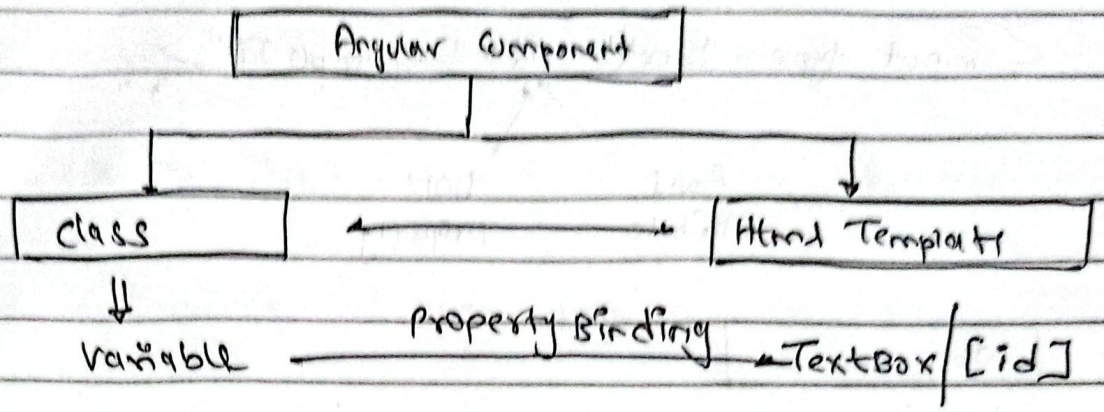
⇒ [abc]
text Box

Console → $0.getAttribute('value') → 'abc'
$0.value → 'abc'

Changed → [xyz]

Console → $0.getAttribute → 'abc'
$0.value → 'xyz'

Attribute (constant)	Property
Attribute are defined by HTML	properties are defined by the DOM
The value of an attribute is constant	The value of a property is changeable
Attribute are used to initialize the DOM properties. After initialization, the job is finish	No such job defined



Defination

Property binding is a one way binding techique, which will help to bind values of the properties of Html elements.

```
<input [value] = "inputText" >
```

property binding

11) Class Binding

PAGE No.
DATE / /

⇒ Angular class binding is used to add or remove classes to and from the HTML elements

⇒ you can add CSS classes conditionally to an element, hence creating a dynamically styled elements

⇒ The Angular provides the three ways to add/remove class to and from the elements.

→ One using the DOM ClassName Property.
→ The second option is to use the class shorthand.
→ The third option is to use the NgClass directive.

① Class binding with ClassName :-

The ClassName is the property name of HTML Element. Hence we can make use of Property binding to assign the class name to any Html element.

< div [className] = " is Special" > < div>

② ..

③ ...

⇒ Conditionally apply classes

- we can also bind the class name dynamically.
- To do that first create a variable in your component class and then use it in Html template.
- you can create a function, which returns the class.

12) # Angular ngClass Directive

PAGE No
DATE / /

⇒ **What is Directive ?**

In Angular, Directive are defined as classes that can add new behavior to the elements in the template or modify existing behavior

⇒ The purpose of Directive in Angular is to work with the DOM, be it by adding new elements to DOM or removing elements and even changing the appearence of the DOM elements.

⇒ The Angular ngClass Directive is an Angular Attribute Directive, which allows us to add or remove CSS classes to an HTML element.

⇒ Using ngClass you can create dynamic style in angular components by using conditional expressions

⇒ **NgClass with a String**

→ you can use the string as expression and bind it to directly to the ngClass attribute.
→ If you want to assign multiple classes, then separate each class with space

⇒ **NgClass with Array**

→ You can achieve the same result by using an array instead of a string

PAGE No.
DATE / /

⇒ **NgClass with object**

- you can also bind the ngClass to an object. Each property name of the object acts as a class name and is applied to the element if it is true

⇒ **Dynamically updating class names**

→ we can dynamically change the CSS classes from the compon..
- using strings
- using arrays
- using Javascript object
-

⇒ we can also use the ngClass without a square bracket. In that case, the expression is not evaluated but assigned directly to the class attribute. we also need to remove the double quote around the expression.

13) Style Binding / ngStyle Directive

PAGE No.
DATE / /

⇒ To create a single style binding, use the below :..

[style.width] = "width"

⇒ Binding to multiple styles :-

[style] = "Style Expression".

→ A string list of ~~syt~~ style such as
"width: 100px ; height: 100px; ..... ;".

→ An object style name as keys and style values
{ width: '100px', height: '100px', ... }.

<u>Note</u> :- that binding an array to [style] is not supported.

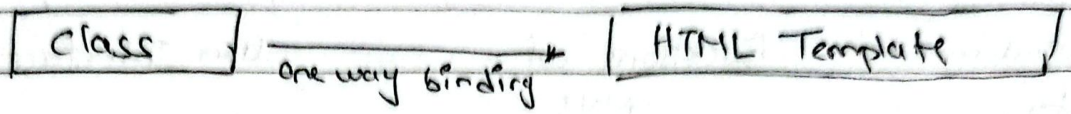
⇒ we can't add styles conditionally

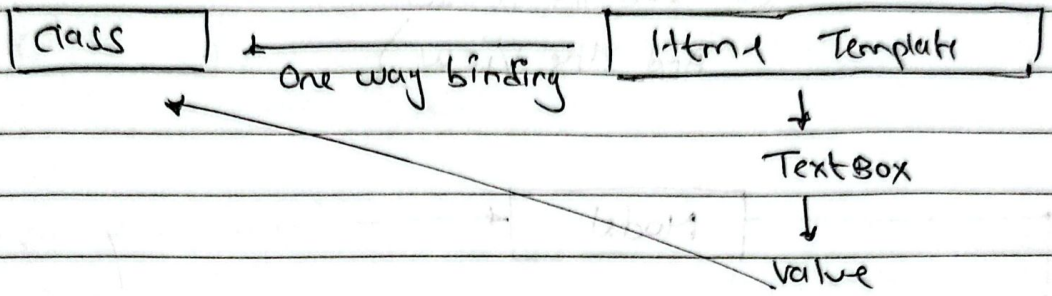
⇒ we can use ngStyle Directive in place of [style].

14) Getting TextBox value / Template Reference variable (# input Ref)

PAGE No
DATE / /

Binding In Angular





⇒ The Template reference variable is a reference to any DOM element, component or a directive in the Template.

⇒ we can use it elsewhere in the template

⇒ we can also pass it to a method in the component. It can contain a reference to element to elements like input, h1, div, etc.

⇒ we declare Template reference variable using # followed by the name of the variable (# variable)

⇒ We can get any property value by using Template Reference variable e.g :- value, name, placeholder, etc.

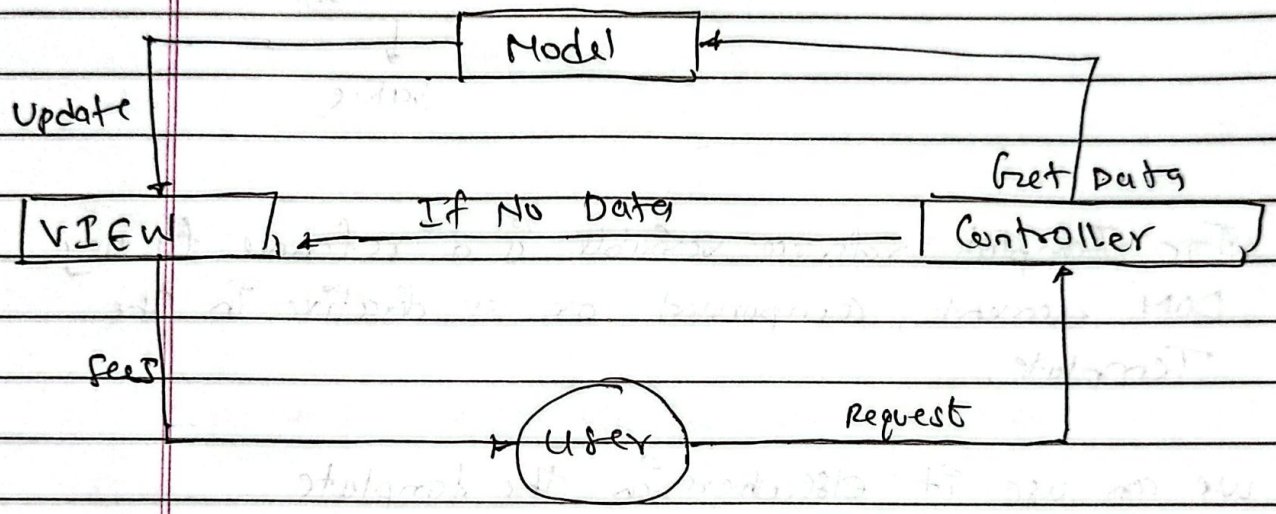
⇒ we can refer to any DOM elements by using

⇒

16) MVC in Angular

Model ⇄ Controller ⇄ VIEW

Application Data — Backbone of MVC (Controls the model And the View) — User Interface



Class → Controller

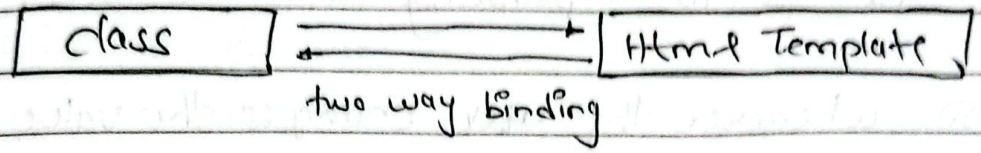
Model class / Properties/function (in .ts file) → Model

Html Template → view

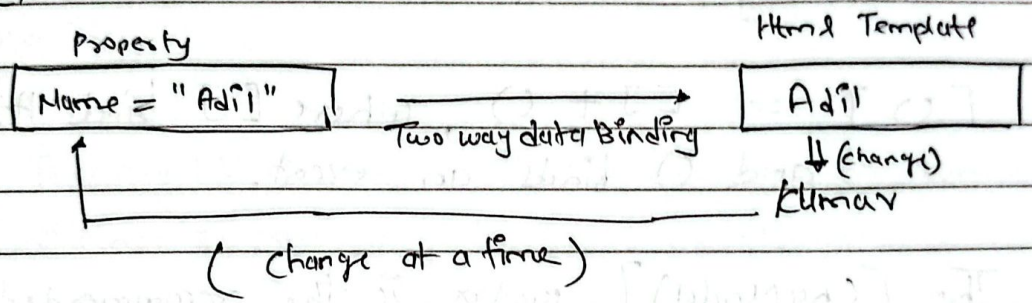
⇒ Every Component is MVC based component

17) Two way Binding

⇒ There are 2 types of binding
→ One way Binding
→ two way Binding



Ex :-



* ⇒ we need to use [(ngModel)] directive
→ import in app.module.ts
import { FormsModule } from '@angular/forms';

```
imports : [
    FormsModule,
],
```

⇒ Two way data binding refers to sharing data between a component class and its HTML template.

⇒ If you change data in one place, it will automatically reflate at the other end.

PAGE No.
DATE / /

⇒ ngModel Directive

→ The ngModel directive with [( )] syntex (also known as banana box syntex) syncs value from the UI to a property and vice-versa

→ So, whenever the user changes the value on UI, the corresponding property value will get automatically updated.

→ [( )] = [ ] + ( ), where [ ] binds Attributes, and ( ) binds an event.

- The [(ngModel)] syntex is the recommended way of two-way data binding

⇒ Important Note

→ If you are using ngModel in your angular app then you have to add below line inside app.module.ts file.

```
import { FormsModule } from '@angular/forms';
```

18) 19) ngIf Directive, ngSwitch

PAGE No.
DATE / /

⇒ Stuctural directives are responsible for the structure and Layout of the DOM Element. It is used to hide or display the things on the DOM.

- ngIf
- ngIf-else
- ngSwitch
- ngfor

⇒ ngIf

- The ngIf Directive is used to add or remove HtmL Elements according to the expression.

* - The expression must return a Boolean value.

- If the expression is false then the elements is removed, otherwise element is inserted. (also in Elements section in browser)

- The ngIf directive does not hide the DOM element. It removes the entire element along with its subtree (child elements) from the DOM.

⇒ ngIf-else

- ngIf-else works like a simple If-else statement, where in if the condition is true the 'If' DOM element is rendered, else the other DOM Element is rendered.
- Angular uses 'ng-template' with element selector in order to display the else section on DOM

PAGE No.
DATE / /

⟹ **ngSwitch** :- It is similar to switch statement of any other programming langu.

→ The ngSwitch directive is applied to the container element with a swith expression.

⟹ **ngSwitchCase**

→ In ngSwitchCase directive, the inner elements are placed inside the container element.

→ The ngSwitchCase directive is applied to the inner elements with a match expression.

→ Whenever the value of the match expression matches the value of the switch expression, the corresponding inner element is added to the DOM. All other inner elements are removed from the DOM.

→ If there is more than one match, then all the maching elements are added to the DOM.

⟹ **ngSwitchDefault**

→ The element with ngSwitchDefault is displayed only if no match is found.

→ If you add more than one ngSwitchDefault directive; all of them are displayed.

PAGE No
DATE / /

⇒ **ngFor**

Anglar ngfor directive iterates over a collection of data like an array, list, etc, and creates an Html element for each of the items from an Html Template

⇒ Nested-ngfor

22) How to install and use Bootstrap

PAGE No.
DATE / /

⇒ Install bootstrap in
- -npm install bootstrap (or)
- -npm install bootstrap@3.3.7
  " @4

⇒ latest version: "5.3.2".

[Note:- If bootstrap version is below 5 then we need to install jQuery separately
- -npm install jQuery]

⇒ ~~Register~~
You can check bootstrap installed in package.json file inside your angular app

⇒ Now open your angular.json file and register bootstrap and jQuery files over there
→ Add below line in "Styles" array inside your angular.json file.

"node_modules/bootstrap/dist/css/bootstrap.min.css"

→ Add below lines in "scripts" array inside your angular.json file

"node_modules/jquery/dist/jquery.min.js",

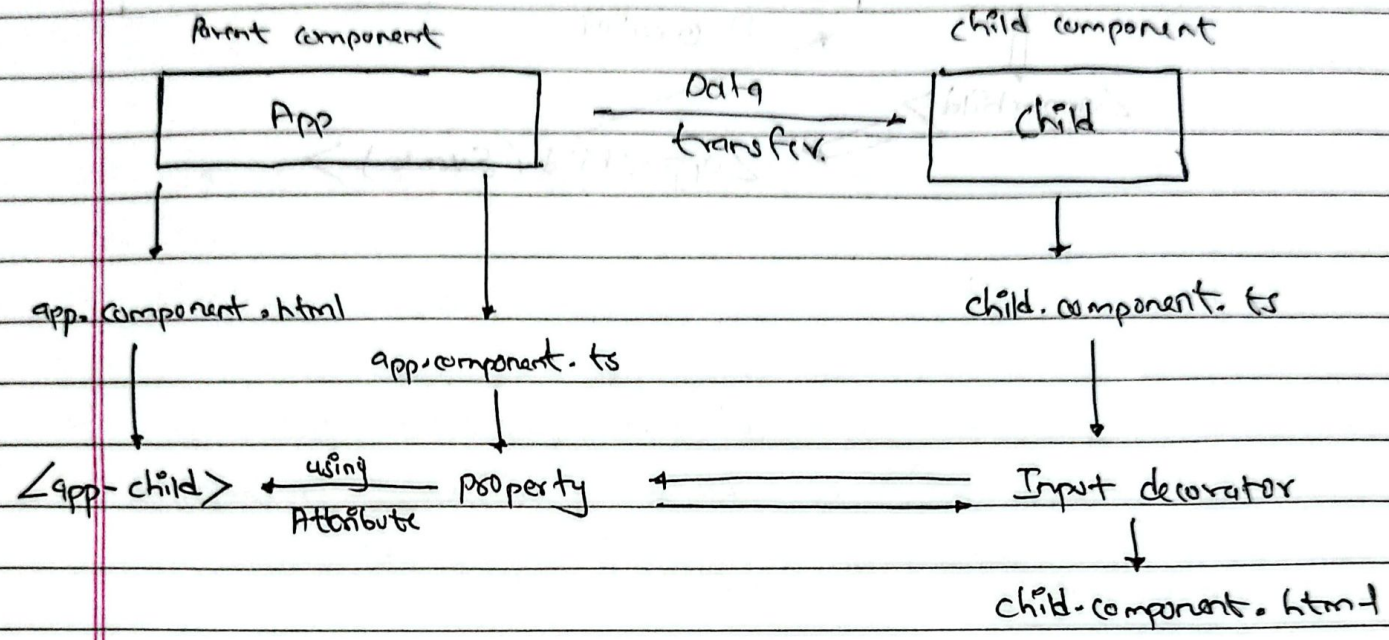
"node_modules/bootstrap/dist/js/bootstrap.min.js"

→ now these file are avaible in whole angular application

23) How to Pass Data Parent to child component

PAGE No
DATE / /

Passing Data to child



⇒ In child class.ts
import { Input } from '@angular/core';

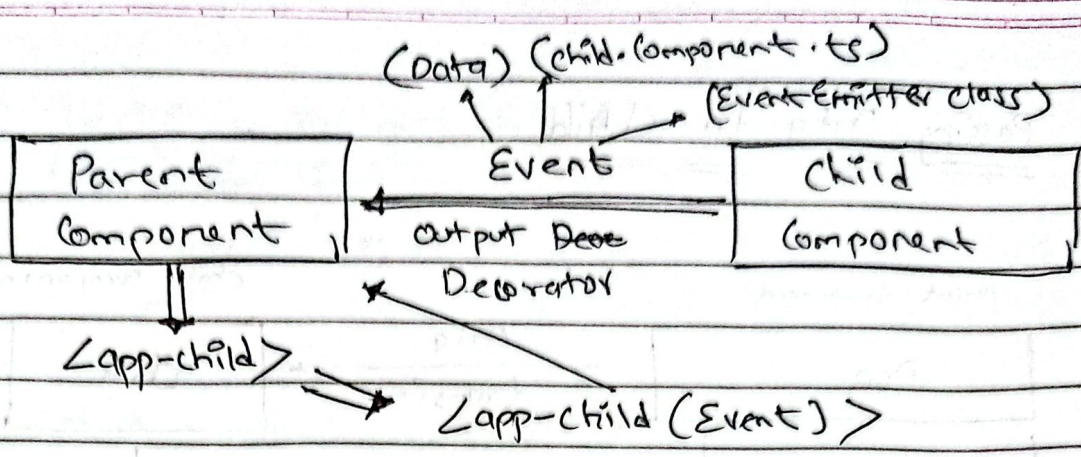
⇒ the Parent Component can communicate with the child component by setting its Propery.

⇒ The child Component does this by using the property decorator '@Input'.

⇒ **In the Child Component**

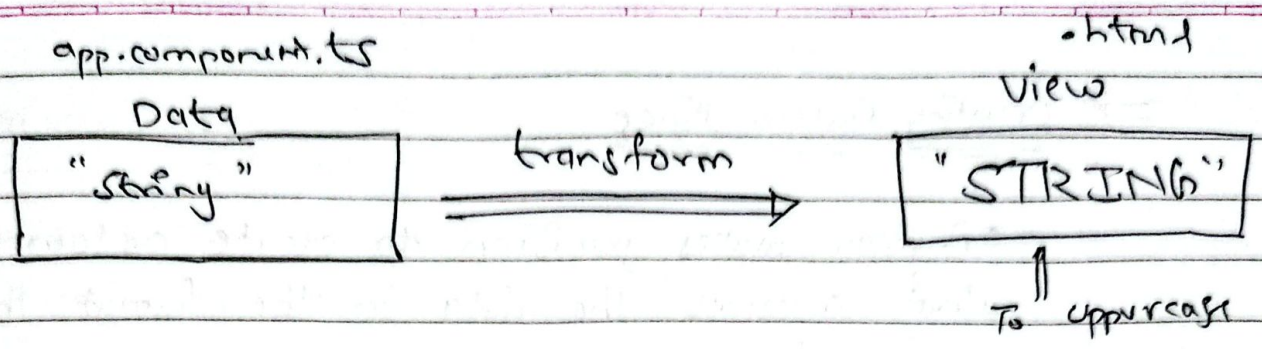
→ Import the @Input module from @angular/core Library
→ This decorator as also support change tracking.
→ When you mark a property as input property, then the Angular injects values into the component property using property Binding

24) Passing Data from child To Parent component



25)
26)
27)

Pipes In Angular



⇒ It is used to transform data.

⇒ Symbol "|"

⇒ Syntax :- {{ Welcome | lowercase }}

⇒ It takes integers, string, arrays, and date as input separated with | to be converted in the format as requered and display the same in the browser

⇒ Types of pipes

→ lowercase pipe, uppercase, titlecase pipe, slice pipe, json pipe, date pipe, currency pipe, percent pipe, decimal pipe.

28) Creating custom pipes

PAGE No.
DATE / /

⇒ Creating Custom Pipes

→ Angular makes provision to create custom pipes that convert the data in the format that you desire.

→ Angular pipes are TypeScript classes with the @Pipe decorator

→ The decorator has a name property in its metadata that specifies the pipe and how and where it is used

⇒ Command to creat custom pipe

- ng g p pipe_name
(or)
- ng g p folder/pipe_name

29) Why do we need service in Angular
30) Services & Dependency Injection

PAGE No.
DATE / /

⇒ Angular Service

* → Service is nothing just a class with specific purpose.
→ Data sharing Across multiple components.
→ It is used to apply application logic
→ we can use services for external interaction such as connecting to a data-source/database.

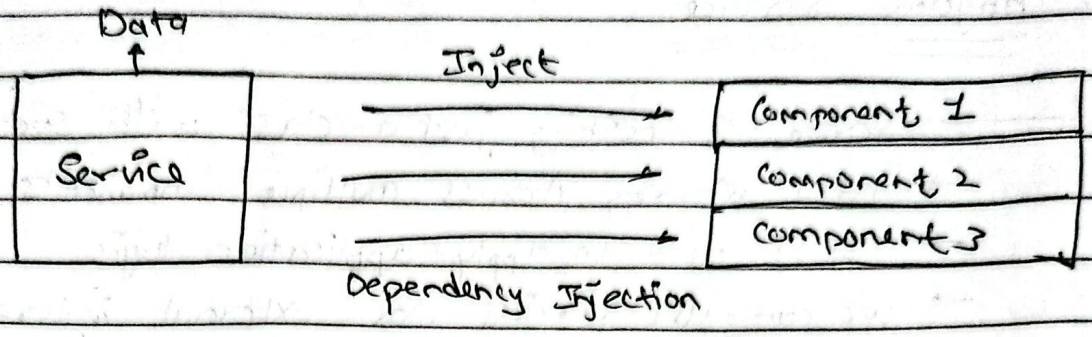
* → The best solution is to write services and inject in application where we need it
→ Services are usually implemented through dependency injection

⇒ The Entire UI of the Angular application is devided into several component

→ Sometimes all these components performing common tasks like displaying images on the view, accessing the database etc
→ To avoid writing code again and again, Angular service comes into place.

→ These service can then be injected into the component that require that service or dependency.

Service & Dependency injection



⇒ Service

→ The main objective of a service is to organize and share business logic, model, or data and functions with different components of an Angular application

→ Services are piece of code that are used to perform a specific task, a service can contain a value or function or combination of both.

→ The best solution is to write services and inject in application where we need it.

→ Services are usually implemented through dependency injection.

⇒ Naming Convension for service

- fileName . service . ts

- ng g s services / StudentService

PAGE No.
DATE / /

⇒ Scenario Without DI

```
class Engine{
    constructor () {}
}
class Tires {
    constructor () {}
}
```

```
class car {
    Engine;
    tires;
    constructor () {
        this.engine = new Engine ();
        this. tires = new Tires ();
    }
```

→ initializing object everytime
→ tightly coupled.
→ car dependent on Engine and Tires

⇒ DI as a Design Pattern

→ Dependency Injection is a coding pattern in which a class receives its dependencies from external sources rather than creating them itself.

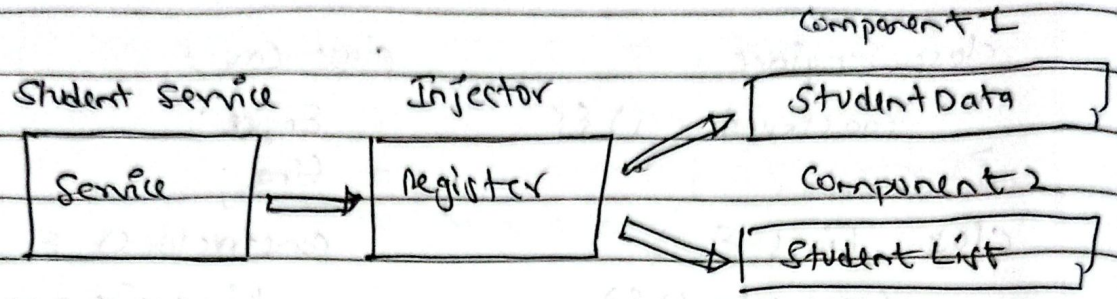
(with DI)
(constructor injection)

```
class car {
    engine;
    tires;
    constructor (engine, tires) {
        this. engine = engine;
        this. tires = tires;
    }
}
```

```
var myEngine = new Engine ();
var myTires = new Tires ();
var myCar = new car (myEngine, myTires);
```

PAGE No.
DATE / /

⇒ DI In Angular

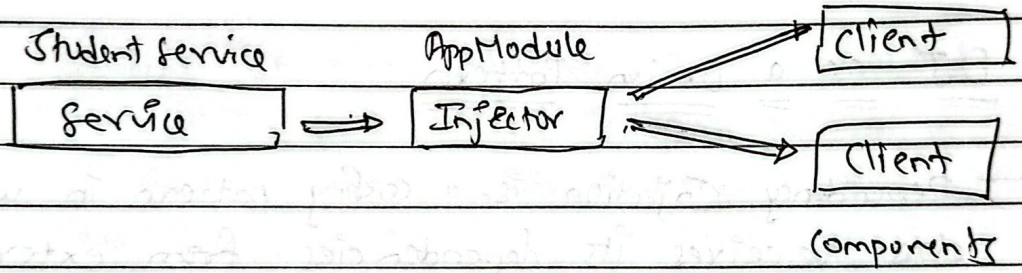


Steps :-

→ Create Student Service service class.

→ Register with Injector

→ Add the Student-Service / dependency in All components where needed.

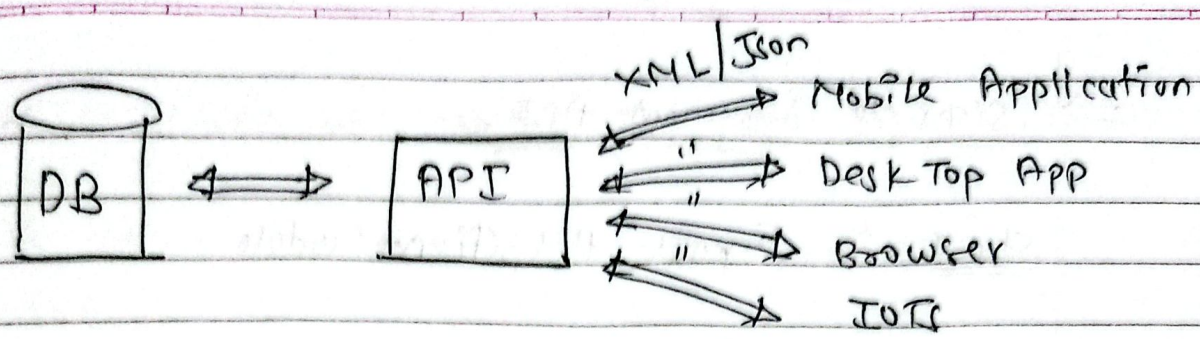


⇒ 2 way to register service

→ 1. App level : If you want the service in your all components then register service at App level.

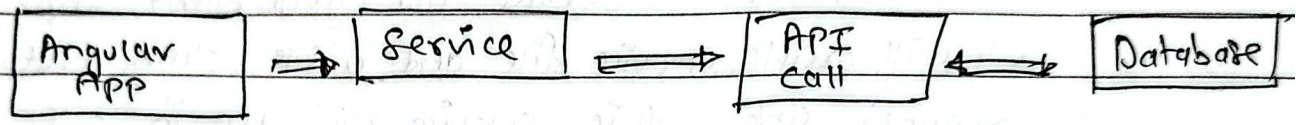
→ 2. Component level : If you want the service in your particular component Then register service with component level.

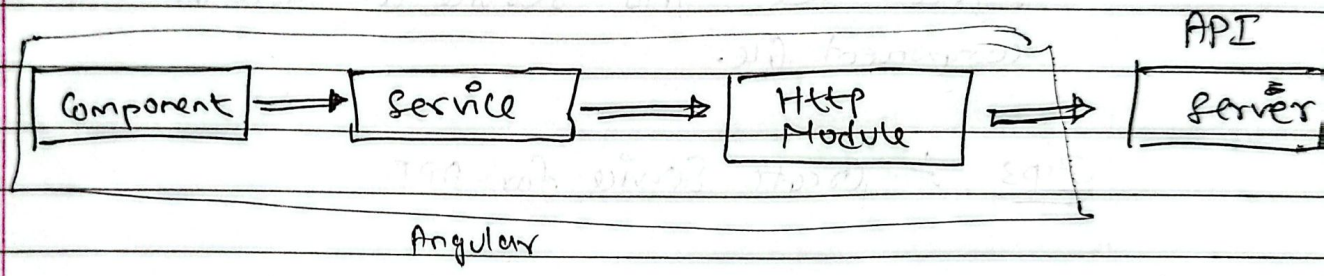
32) Calling API In Angular
Get Data & Display

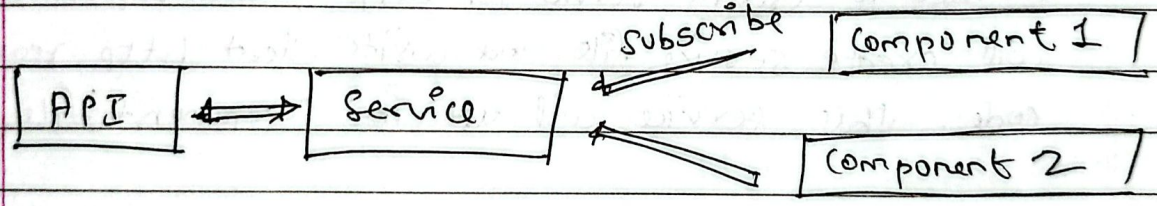


Http protocole / verbs (Get, Post, Put, Delete)

⇒ Getting API Data In Angular







⇒ Every APP must important to sending API request to another server. we can use http client request and get data and store data information to our server

⇒ Specially when you are working with angular, ~~vue~~ vue, react application, you must have to learn how to run http client request.

PAGE No.
DATE / /

⇒ Steps to Implement API

Step 1 :- Import HttpClient Module

we need to import HttpClientModule to app.module.~~ts~~ file. ~~So~~

Step 2 :- Create service for API

we need to create service for http client request. we will create service file and write client Http request code. This service will use in our component file.

Step3 :- Create Service for API

we need to create service for http client request. we will create service file and write client http request code. This service will use in component file.

Step 4 :- Use service to component.

34) Angular forms

PAGE No.
DATE / /

⇒ Angular forms are used to handle user's Input

⇒ We can use Angular form in our application to enable users to login, to update profile, to enter information, and to perform many other data-entry tasks

⇒ In Angular, there are 2 approaches/types to handle user's input through form
- → Template-driven forms
- — Reactive forms (Model Driven Forms)

⇒ Both approaches are used to collect user Input events from the views, validate the user input, create a form model and data model to update, and provide a way to track changes.

⇒ <u>Template Driven Forms</u>

→ Best if you want to add a simple form to your application (Ex:- login, contact, signup form)

→ Easy to use in the application but they are not as scalable (expandable/upgradable) as Reactive forms.

→ Template driven forms is created using directives in the template. Ex: ngForm, ngModel

→ Basic HTML validations can be used to validate the form fields, ex:- required, maxlength, pattern.

PAGE No.
DATE / /

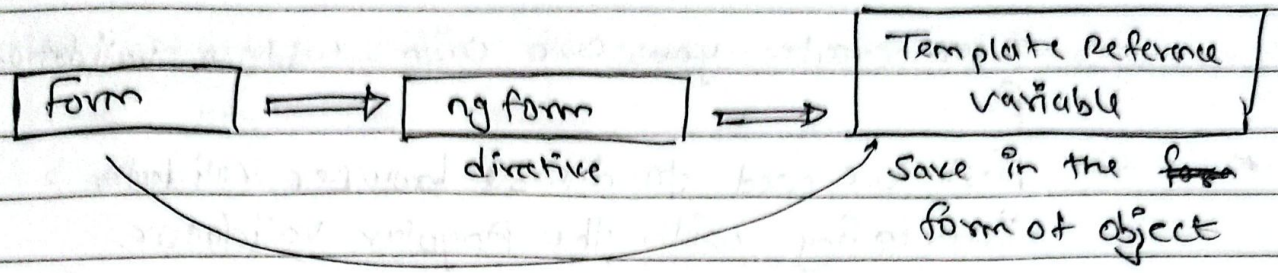
- In the case of custom validations, directives can be used.
- Template-driven forms are less explicit.
- They are created in HtML template.
- It works like Asynchronously

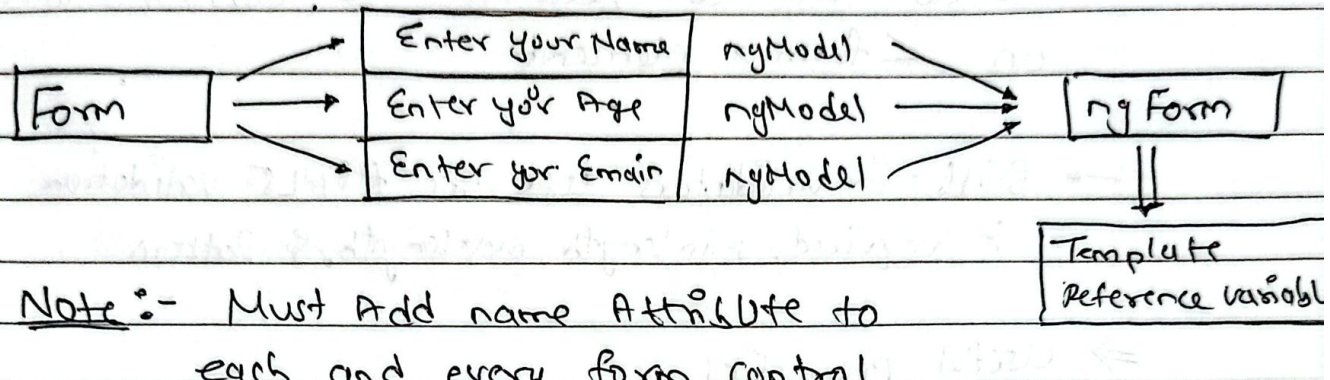
⟹ **Reactive Forms**

- Reactive forms are more robust (powerful)
- They are created in Component class.
- Reactive forms are more scalable, reusable, testable
- They are most preffered to use if forms are a key part of your application
- Reactive forms are more explicit (manual)
- It works like synchronously.
- Reactive forms are code-driven, unlike the template-driven approach.
- Reactive forms eliminate the anti-pattern of updating the data model via Two-way data binding

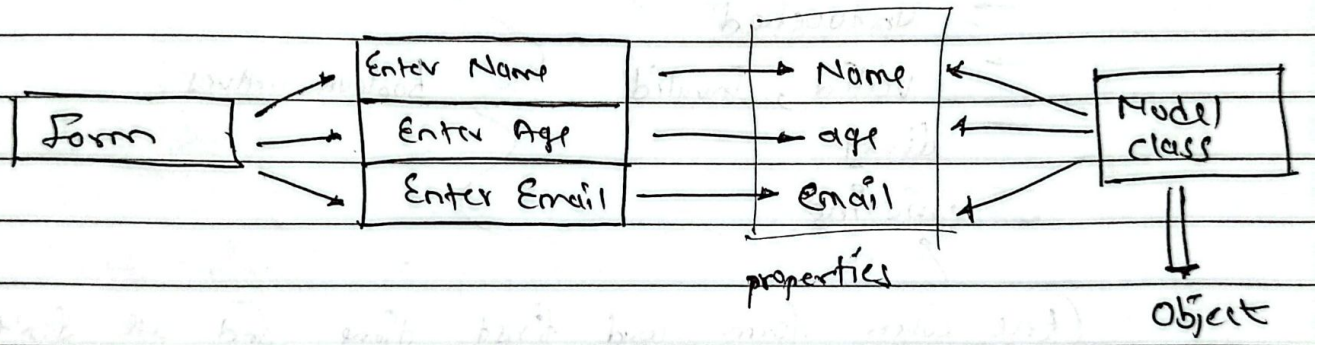
35) to 40)

Template Driven forms





Note :- Must Add name Attribute to each and every form control.



Two way data Binding

⟹ Form validation

→ validation in template-driven forms are provided by the validation directives.

→ The Angular forms Module come with several built-in validation.

PAGE NO.
DATE / /

→ we can create you our own custom validation

* → First we need to disable browser validator interfering with the Angular validator.

→ To do that we need to add 'novalidate' attribute on <form> element

→ Built-in validators use the HTML5 validation i- required, minlength, maxlength & Pattern

⇒ Useful Properties
- → touched
- → untouched
- → valid, Invalid
- → dirty
- → pristine

} boolean values

(true when form load first time and we don't change any-thing in form or field)

(dirty is opposite of pristine)

⇒ "is-invalid" class gives red border when field is empty with required validation
(in Input class)

opposide
"is-valid"

[see 'name' field in html in project for validation]

PAGE No.
DATE / /

⇒ Submit button is disable when any ~~error~~ occur or any blank field
employee form

→ in submit button tag
[disabled] = "employeeForm.invalid"

⇒ When form load All field should be blank so put nullable all field in Model.ts class

⇒ Radio Buttons & DropDownList

⇒ CheckBox

⇒ Reset form

41) to 45)

Reactive forms In Angular

Template Driven form	Reactive form
→ Simple Basic forms	→ complex form with more control
→ Easy to start with	
→ Based on HTML template	→ Structure of forms defined in TypeScript class

⇒ **Reactive forms**

app.component.Html ⇓ HTML Template ⇓ Form

app.component.ts ⇓ class ⇓ FormGroup (obj that represent form)

Bind ⟷

Enter Name	Bind ⟷	name → FormControl
Enter Age	Bind ⟷	age → FormControl
Enter Email	Bind ⟷	email → FormControl

PAGE No.
DATE / /

⇒ Introduction

→ It is also known as model-driven-forms

* → With template-driven forms, 'template-directives' are used to build an internal representation of the form.

* → With reactive forms, you build your own representation of a form in the component class.

→ There is no need of Two way Binding.

Reactive forms

→ Reactive forms are forms where we define the structure of the form in the component class, i.e, we create the form model with formGroup, form controls and Form Array
  ↳ Then, we bind it to the Html form in the template

→ we also define the validation rules in the component class.

→ This is different from the template-driven forms, where we define the logic and control in the HTML template

⇒ Here are some of the advantages of reactive forms:

- → Using custom validators
- → Changing validation dynamically
- → Dynamically adding form fields.

⇒ **How to use Reactive forms**

1. Import ReactiveFormsModule
2. Create forms Model in component class using FormGroup, FormControl & FormArray
3. Create the html Form resembling the form Model
4. Bind the html Form to the Form Model.

⇒ **Form Validation**

46) What is Directive & How to Create custom Directive

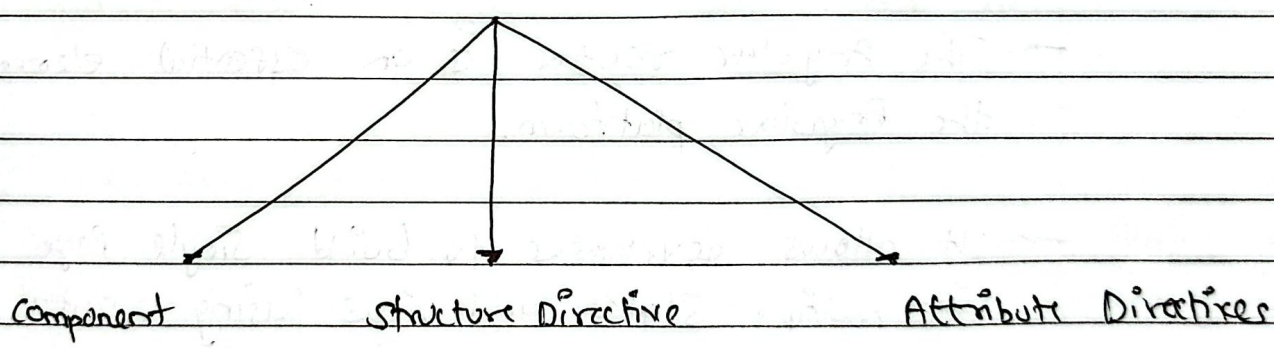
PAGE No
DATE / /

⇒ Directive are defined as classes that can add new behavior to the elements in the template or modify existing behavior.

⇒ The purpose of Directive in Angular is to Manipulate the DOM, be it by adding new element to DOM or removing elements and even changing the appearance of the DOM elements.

⇒ Directive are meant to be a function that executes when found in the DOM by the Angular compiler to extend the power of the HTML with new syntax.

⇒ <u>Types of Directives in Angular</u>



⇒ Custom Directive is very similar to creating the Angular component. The custom directive is created using the @Directive to replace the @Component decorator.

• ng g directive name

47) Introduction To Routing

PAGE No.
DATE / /

⇒ What Is Routing

→ Routing is the mechanism by which request (as specified by a URL and HTTP method) are routed to the code that handles them

URL

[.com/Index] → Code Executes → Index

// / About → // ← About.

→ using Angular Router

⇒ Angular Router

→ The Angular router is an essential element of the Angular platform.

→ It allows developers to build Single Page Appl... with multiple states and views using routes and components and allows client side navigation and ~~routi~~ routing between the various components.

→ It's built and maintained by the core team behind Angular developement

→ @~~Angto~~ angular / router package

48) Configuring Routes

PAGE No.
DATE / /

Steps :-

→ Step 1 :- Create a New Angular Project with Routing

[→ How to Add Routing to an Existing Angular Project
• ng generate module app-routing --flat --module==app]

→ Step 2 :- Create components then configure in app-routing.modules.ts.

<router-outlet> </router-outlet>

→ render clicked components at this place

→ Step 3 :- Create Routes in Routing class file.

⇒ The Router-Outlet is a directive exported by 'RouterModule' and acts as a placeholder that indicates to the router where it needs to insert the matched component(s).

⇒ A collection of routes defines the router configuration which is an instance of Routes.

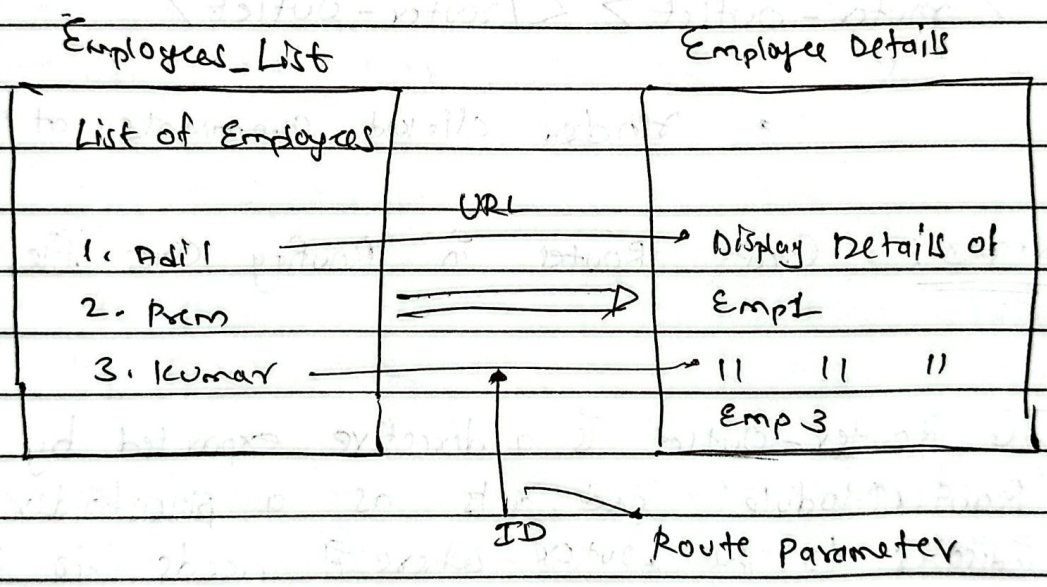
49) Page Not found, wildcards & path Match in Angular

→ {path: '', redirectTo: '/home', path: 'full'};

{path: '**', component: pageNotfoundComponent}

wildcard : if No url found goto not fond page

50) Route Parameter in Angular / Dynamic Routing



51) Multiple Route Parameter

52) ParamMap Observable in Angular Routing

PAGE NO.
DATE / /

⇒ ParamMap is the method of ActivatedRoute and it will return an observable and it will give data only if we subscribe to it.

⇒ The argument in the subscribe method is an arrow function.

⇒ The observable provides the parameter which we strongly typed to ParamMap

⇒ ParamMap is comming from Router package

53) Optional Route Parameters in Angular Routing

Absolute & Relative Path

/About → Absolute Path

About → Relative Path

⇒

Home → About → Home

↓

Link → Home → Relative Path

↓

"Home" → without Forward Slash/

(root URL)

localhost : 4200/About/Home (X → not correct URL)

⇓ To overcome this issue use Absolute path

→ Absolute path append with root URL

→ Practical in 'interpolation ~~URL~~ route' in routing.Html.

★ Note:- ~~Abs~~ Absolute Path always concatinate with root URL (localhost : 4200)
result :- localhost : 4200/Home

→ Relative path concatinate with current URL
result :- localhost : 4200/About/Home

55) Child Routes / Nested Routes

PAGE No.
DATE / /

Home
About
Products } links

All Products Display Here

T-shirts | Trouser

All Trousers Display Here.

(Child Components, Child Routes)

56) Implementing All The Routes Inside Bootstrap Navbar

PAGE No.
DATE / /

58)

57) Auxiliary (extra) Routes / Named Routes & Multiple Router Outlets.

(extra router outlet)

Home About Contact Course Course-Details

Home
About
Course
} <router-outlet>
(primary router outlet)

Courses Details
1. Python.
2. Java
3. C#
} <router-outlet name = 'r1'>

⇒ Angular supports the concept of auxiliary routes, which allow yo to set up and navigate multiple independent route in a single app

⇒ Auxiliary routes allow the user to access or toggle portion of the page, such as a side-bar or dialog, using the URL.

⇒ Each component has one primary route and zero or more auxiliary outlets.

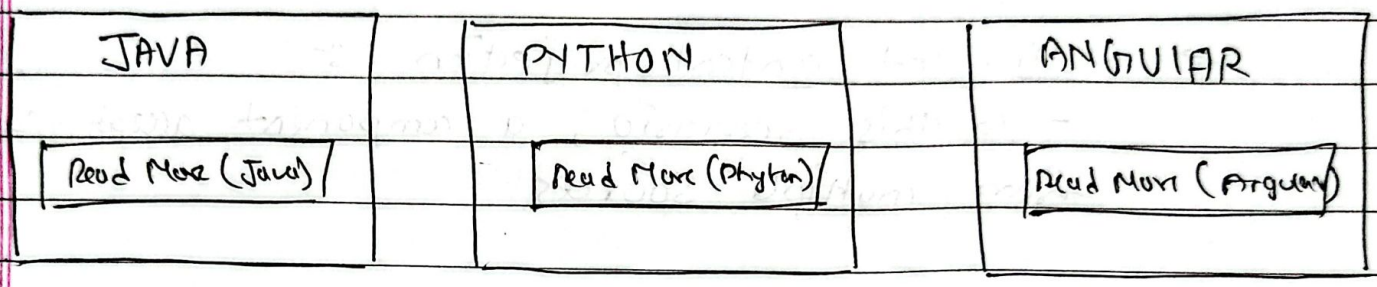
⇒ Auxiliary outlets must have unique name within component

⇒ To define the auxiliary route we must first add a named router outlet where contents of the auxiliary route are to be rendered.

58) ng-content

⇒ The ng-content is used when we want to insert the content dynamically inside the component that helps to increase component reusability

⇒ Scenario



⇒ Using ng-content we can pass content inside the component selector and when angular parses that content that appears at the place of ng-content

⇒ Content Projection

→ Content projection is a pattern in which you insert, or project, the content you want to use inside another component.

→ For Example :- you could have a card component that ~~can~~ accepts content provided by another component.

PAGE No.
DATE / /

Types of Content Project

→ Single - slot content projection :-
- with this type of content projection, a component accepts content from a single source.

→ Multi - slot content projection :-
- In this scenario, a component accept content from multiple sources

→ Conditional content projection :-
- Components that use conditional content projection render content only when specific conditions are met