What is angular?

Angular is a platform and framework for building single page client application. using HTML, CSS and type script. Angular is written in Type Script.

* Angular is developed and created by Google.

What are the features of Angular?

① Cross Platform :

With angular we develop progressive web application (PWA). Angular application is suitable for running in multiple platforms.

② Code Generation :

Angular has auto code generation process that is it turns your template into code.

③ Code splitting:
Angular apps load quickly with the new component router, which delivers automatic code splitting so that users only load code that they want to see.

④ Angular CLI:
Angular uses command line tools to start building fast, add more components or tests and then we can deploy the application instantenously.
(CLI → Command Line Interface)

⑤ Provides Templates:
Quickly creates UI views with simple and powerfull template syntax.
When angular project is first created then by default angular provides default templates.

(UI → User Interface).

⑥ Testing : (karma & Protector)

With karma for unit tests, you can know if you have broken things everytime you save.

Protactor makes your scenario test ~~runs~~ to run faster and stable manner.

Why angular over other framework?

① MVC Architecture Implementation:

(MVC → Model View Controller)

The controller receives all the requests for the app and operates with model to prepare any data needed by the view.

② Enhanced Design Architecture:

The architecture is built in such a way that it helps programmer to ~~load~~ locate and develop the code easily.

③ Modules:

A module is a mechanism that groups directives, components, pipes and services in such a way that they can be combined with others to create and an application.

④ Services and Dependency Injection:

* We will ellaborate about this topic later.

⑤ Custom Directives:

Custom directives improve HTML functionality and are suitable for dynamic client side application.

⑥ Angular uses Type Script:

Angular is written using Type-Script which is superset of Java-Script.
The above line means that it not only compiles Java Script but also helps programme to spot and eliminate common mistakes while coding.

Difference between Angular and Angular JS

Angular JS	Angular
① Written in Java Script	① Written in Type Script.
② Is not mobile friendly	② Is mobile friendly app.
③ special methods and {{}} are used for data binding.	③ Only () and [] are used for data binding.
④ Dependency Injection is not used.	④ Hirarchical DI is used in angular.
⑤ Angular JS is difficult to manage with increasing the size of the source code.	⑤ Angular code is better structured, is easy to create and manage bigger application.

What is Dependency Injection?

DI is wired into the Angular framework and used everywhere to provide components with the services or other things they need.
It is a design pattern.
We check DI in ngDoCheck() function of Angular life hook.

When does component in Angular gets destroyed?

When the angular application moves from one component to another component and the previous component is no longer needed then we that previous component gets destroyed.

Ex: In FB, first we go to login page to login to the user account, now once the login is completed succesfully then the login page is no longer needed. So the component of the login page gets destroyed.

Life Cycle Hook of Angular:

1. ngOnChanges()
2. ngOnInit()
3. ngDoCheck()
4. ngAfterContentInit()
5. ngAfterContentChecked()
6. ngAfterViewInit()
7. ngAfterViewChecked()
8. ngOnDestroy()

1. ngOnChanges():

This is used for detecting changes in input properties.

Responds when Angular sets or resets data bound input properties.

It is called before ngOnInit(), whenever one or more data bound input properties change.

Note: If component has NO INPUT then framework will not call ~~ngOnInit()~~. ngOnChanges().

② ngOnInit():

Initialises the component after Angular first displays the data bound properties and sets the component's input properties. Basically initialises Angular components after inputs are received.

③ ngDoCheck():

Detects and acts on the changes that Angular can't detect on its own.

④ ngAfterContentInit():

Respond after Angular projects external content into the component's view.

(It is actually used when we initialize every variable on the array or object in ngOnInit) or (we perform login in the above three life cycle hooks and we want to execute one a function.) or (to make some calculation one time in the whole component irrespective of data change or two way binding).

⑤ ngAfterContentChecked():

Respond after Angular checks the content projected into the directive or content.
This will check whether the content has been loaded or projected.

⑥ ngAfterViewInit():

Respond after Angular initializes the component's views and child's view or the view that contains the directive.

This will basically check the Angular applications view.

⑦ ngAfterViewChecked():

Respond after Angular checks the component's view and child's view.

Basically this will check the view that will not be checked by Angular.

⑧ ngOnDestroy():

Cleanup just before Angular destroys the directive or component.
Basically unsubscribe observable to avoid memory leaks. Also it checks what data is destroyed.

Navigation / Routing Config:

Routing is basically means navigating between pages. It also tells how to go from one component to another in angular application.

Why Angular has AOT compiler?

So basically we have two types of compiler
1. AOT (Ahead of Time)
2. JIT (Just in Time)

1. Ahead of Time:

The angular AOT compiler converts your Angular HTML and TS code into efficient JS code during the build phase before the browser run the code.

Benefits of AOT:
1. Faster rendering (faster compiling time)
2. Fewer asynchronous requests
3. Detect template errors earlier
4. Better security

(2) Just In Time:

This compiler will start compiling as soon as you save the code and run it.

JIT is used in spring boot application.

What are services in Angular?

It is a common data storage place (temporary) where it stores data from server and provides that data to different components simultaneously.

(*) Angular services are singleton object that gets initialized only once during the lifetime of an application.

The main objective of a service is to organise and share logic, model or data with different components of an angular application.

Ex: If we are logged in in whatsapp in mobile and laptop both then when one message comes in that message is delivered in both the places without creating duplication.

when once a component is destroyed then the data stored in the destroyed component is stored in services.

what is Bootstrapping in Angular?

Bootstrapping is a technique ~~in Angular~~ of initializing or loading our Angular application.

It is where the application is loaded and angular comes to life.

The angular takes the following steps to load our first view:

1. Index.html loads
2. Angular, libraries and application loads
3. Main.ts (application entry point)
4. Root module. (app module)
5. Root component (app component)
6. Template. (app.component.html)

What is string interpolation?

String interpolation is a one way data binding technique which is used to output the data from type script code to HTML template and vice versa.

{{ data }} → syntax.

What is data binding in Angular?

It allows us to define communication between the component and view.

Types of Data binding:

1. String Interpolation
2. Property Binding
3. Event Binding
4. Two way Binding

2. Property Binding:

Property binding in simple term is defined as updating the value of a certain variable in component and displaying it in view.

(3) Event Binding:

Event binding is defined as updating of the value of a certain variable from presentation to the component. Ex: clicking a button.

(4) Two way Binding

Two way binding is a combination of both property binding and event binding.

It is a continuous synchronization of a data from view to the component and vice versa.

Ex → Google traslate

What is a selector in Angular?

A selector is used to identify each component uniquely into the component tree and it also defines how the current component is represented in the HTML DOM.

selector is present in ts file as

- @component({
  . . . . . .
  . . . . . })

Observables and Promises:

Observables:

Angular makes use of observables as an interface to handle a variety of common asynchronous operations.

Promises:

It is has the same function that of observable but it deals with one asynchronous event at a time.

Observables	Promises
Emit multiple value over a period of time.	Emit single value at a time.
Are lazy: they are not executed until we subscribe to them using subscribe() method.	Are not lazy: execute immediately after creation.

Have subscriptions that can be cancelled using the unsubscribe() method, which stops the listener from receiving further values.	Promises are not cancellable.
It provides operations like: forEach, filter, reduce, retry and retryWhen.	Don't provide any operations
Delivers errors to subscribers	Push errors to the child promise.

What is BOM and DOM and which is used for angular?

DOM (Document Object Model):

It is a cross platform and language independent interface that treats HTML and XML document as a tree structure wherein each node is an object representating a part of document.

The DOM represents a document with a logical tree.

When a web page is loaded, the browser creates a Document Object Model of the page which is an object oriented representation of an HTML document, that acts as an interface between Angular and document itself.

DOM works as a parent child relationship which is called NODE.

BOM (Browser Object Model)

The BOM is a browser specific convention referring to all the objects exposed by the web browser. There is not strict definition. so browser vendors are free to implement the BOM in any way they want.

What are directives?

An angular component is nothing more than a directive with a template.

Angular directives are abo used to extent the power of HTML by giving it a new syntax. It is basically a class in angular. Components and services are all directives. When there is multiple component but has similar functionality then we use directives.

Directives are divided into three types;

1. Component
2. Attribute directive
3. Structural directive.

1. Component :- directives with a template.
2. Structural Directives :- change the DOM layout by adding or removing DOM elements.
3. Attribute Directives :- change the appearance or behaviour of an element, component or another directives.

What is difference between constructor and ngOnInit()?

ngOnInit()	Constructor
① It is one of the Angular life cycle hook method.	① It is a type script feature.
② ngOnInit added to prototype of the class created.	② constructor is a function with the same name of class.
③ Called by Angular	③ Called by JS engine.
④ Invoked by Angular when component is initialized.	④ Automatically called at the time of object creation.
⑤ Actual business logic is performed here.	⑤ Used for injecting dependencies.

What is Reactive programming?

Reactive programming is declarative programming paradigm concerned with data streams and the propagation of change.
(Programming with asynchronous calling of Components)

Ex: In imperative programming $a = b + c$ would mean that a is being assigned the result of b + c when the expression is evaluated, later the value of b and c can change but it will have no effect on the value of a.

In Reactive programming, the value of a is automatically updated whenever the values of b and c change, without the program having re-exculed the statement $a = b + c$.

Reactive programming = streams + operation.

* Note: Streams are just a sequence of values over time.

Asynchronous : loading component parallely.
Synchronous : loading component one after another.

What is RxJs?

RxJS : Reactive extension for Java Script.

It is basically a library that gives us an implementation of observables.

So its a library for composing asynchronous and event based programs by using observable sequence.

How do we use Redux in Angular?

Redux is a library in Angular used to manage and maintain the stages of an application. It helps you to implement one way data flow in your angular application. It is inspired by Flux pattern and it also allows you to understand what is going on in your application in a predictable way.

What are the building blocks of Angular?

1. Module
2. Component
3. Template
4. Directive
5. Routing
6. Services
7. Dependency Injection
8. Data binding

What is metadata in angular?

Meta data is used to decorate a class so that it can configure the expected behaviour of the class.
Ex → annotations.

What is ng-Content in angular?

They are used to create configurable components. This means the components can be configured depending on the need of the user. This is also known as content projection.

Basically it is used to show data as it is as normal HTML component.

What is the difference between Annotation and Decorator in Angular?

In Angular, annotation are used for creating annotation array. They are only meta data set of the class using the reflect metadata library.

Decorators in Angular are a design pattern for seperating decoration or modification of some class without changing the original source code.

What is the advantage of TS over JS?

Type Script provides object oriented programming features. It also supports modules. Type script contains interface. It also supports static typing.

What is digest cycle in angular?

Digest cycle is what Angular triggers when a value in the model or view is changed.
Its a monitering procen. It moniters data binding procen. It tracks the changes in the watch variable.

What is pipes in Angular?

Pipes are simple function (which you can use in template expression) to accept an input and return a transformed value.
It is denoted by symbol "|".

Syntax:

```
{{ title | uppercase }}
```

* Pipe takes integers, strings, arrays and date as input seperated with | : It transforms the data in the format as required and displays the same in the browser.

Default pipes provided by Angular:

1. Currency Pipe
2. Date Pipe
3. Decimal Pipe
4. Json Pipe
5. Lower Case Pipe
6. Upper Case Pipe
7. Percent Pipe
8. Slice Pipe
9. Asyn Pipe

What linting in Angular?

Linting is the procen of running a program that analyses your code for ~~programmira~~ programatic and stylistic ~~error~~ manner.
Angular CLI has built in support for code linting. The default tool used by angular CLI is TSLint.

What is Lazy loading in Angular?

Lazy loading is technique in Angular that allows you to load JS components asynchronously while specific route is activated.

It improves the speed of the application load time by splitting the application into several bundles. Whenever the user navigates through the app, the bundles are loaded as required.

OR

By default, NgModules are ~~ear~~ eagerly loaded, which means as soon as the app loads. even if they are not required ~~imp~~ immediatily.

Here lazy loading comes. It will load NgModules as needed.

What is Polyfields?

It is procces of converting type script into efficient Java script. It also defines how the browser should be loaded.

There are two types of fields:

1. Mandatory field
2. Optional field

What is DOMSanitizer in Angular? (Security Term)

DomSanitizer is a service of Angular that helps to prevent attackers from injecting malicious scripts into web pages. (which is also known as cross-site scripting or XSS.)

What is HTTP interceptor?

Most front end application need to communicate with server over the HTTP protocol, in order to download or upload data and accen other backend services.

Interceptors are a unique type of Angular service that we can implement to intercept the incoming and outgoing HTTP request.
It abo provides authentication. With this we can abo ~~inde~~ ~~ency~~ encrypt out code.

What is shared module?

When a module is shared between two or more modules ~~or~~ or components.

Like we use router to connect between components so router is a shared module.

What is a provide?

A provide is an object declared to Angular so that it can be injected in the constructor of your component, directives and other classes.

This are basically services which you can create and provide.

What is NgIf?

It is a structural directive that conditionally includes a template based on the value of an expression coerced to boolean. When the expression evaluates to true, Angular renders the template provided in the then clause, and when false or null, Angular renders the template provided in else clause.

Syntax:

```
<p *ngIf = " serverCreated; else noServer " >
  server was created ~~serve~~ server name is
  {{ serverName }} </p>
<ng-template #noServer >
  <p> no server was created </p>
</ng-template>
```

property 1 → property 2 ←

```
<p *ngIf = " serverCreated ; else noServer ">
(1) server was created ; sever name is {{ serverName }}
</p>

<ng-template #noServer>
(2) <p> No sever was craled </p>
</ng-template>
```

else condition

variable name

if serverCreated propety is true then line 1 is executed else it will execute the "noSever" templati.

What is ng-style?

An attribute directive that updates styles for the containing HTML element.
The ng-style attribute is used to change or style the multiple properties of Angular.

PTO

Ex

HTML :

ngStyle is a directive

we are doing property binding to the directive

```
<p [ngStyle]="{ backgroundColor : getColor() }">
  {{'server'}} with ID {{ serverId }} is {{ getServerStatus() }}
</p>
```

We are changing styling of the element dynamically by applying ngStyle directly into the element.

TS :

```
getColor() {
    return this.serverStatus === 'online' ? 'green' : 'red';
}
```

ternary operator

So basically the ngStyle in HTML calls the getColor function from HTML and dynamically changes color according the "server status".

Applying CSS classes dynamically with ngClass:

HTML

```
<p [ngStyle]="{backgroundColor: getColor()}"
   [ngClass]="{online: serverStatus === 'online'}">
   .....
</p>
```

variable

↳ class name defined in styles of the component.

TS

```
@Component({
   selector: ....
   templateUrl: ....
  {styles: ['
        .online { color: white; }']
```

online class is defined here.

(*) So whenever the server status is online it will call the "online" css class defined in styles of TS.

What is ngFor?

A structural directive that renders a template for each item in a collection.

HTML

this will loop through all the ele. in the array and assign individual element to the dynamic "server" variable.

```
<app-server *ngFor="let server of servers">
</app-server>
```

this is the property defined here

TS

```
servers = [....];
onServerCreation() {
  .....
  this.servers.push(this.serverName);
  .....
}
```

* let server: creates a local variable name server dynamically which is only available in that template

of servers: this will be the property of the component.

What is ngSwitch?

ngSwitch is a structural directive which is used to Add / Remove DOM element.

HTML

```
<div [ngSwitch] = " value " >
① <p *ngSwitchCase = "5" > value is 5 </p>
② <p *ngSwitchCase = " 10 " > value is 10 </p>
③ <p *ngSwitchDefault > Value is default </p>
```

→ Depending on the value the ngSwitch case will be called.

When no value is mentioned or passed in " value ".

If value in TS file is 5 then line 1 will be executed.

If value in TS file is 10 then line 2 will be executed

and if no value in TS file then line 3 will be executed.

Angular Services (Deep Dive)

Principles: (Disadvantage of NOT using Services)

- Do Not Repeat Yourself (DRY code)
- Single Responsibility Principle.

So, to overcome the above disadvantages we use "SERVICE"

So, what is service?

① A class with a specific purpose.

② **Share data**: We need to share data accross multiple components. So in such a scenario a service will be responsible for providing the data to the components that ask for it.

③ **Implement Application Logic**:

Ex → an employee enters a date of birth but we need to calculate the age, to do that this does not need to view and logic should be reusable code which should be independent of any individual component.

(4) External Interaction:

We use service to connect with database and other backend framework like (STS).

Naming Convension : service-name . service . ts

How do we use service in Angular?

→ "Dependency Injection"

What is Dependency Injection?

① Code without DI

Ex → Lets say class Car which is dependent on two classes Engine and Tyre and both the class are static constructor.

So, if we make any change in the sub class then we need to change the main class again and again.

Also the above process is not suitable for testing as the main class will not accept varities.

So to avoid this proble we use DI as a design pattern.

(2) DI as a design pattern:

Basically DI is a coding pattern in which a class receives its dependencies from external sources rather than creating them itself.

Ex →

Without DI

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

Code with DI :

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

We do not create any dependencies we just consume them, creation of those dependencies is external to this class.

①
```
var myEngine = new Engine();
var myTires = new Tires();
var myCar = new Car(myEngine, myTires);
```

②
```
var myEngine = new Engine(parameter);
var myTires = new Tires();
var myCar = new Car(myEngine, myTires);
```

So basically instead of calling the function directly we now pass objects of that ~~each~~ class which inturn calls the functions of their own. It becomes more flexible. Even if we change the functions ~~still objects~~ of the child class we donot need to modify or change the main class.

But if the no. of dependency grows it really becomes difficult to manage the code.

So to avoid the problem DI as a framework comes into picture.

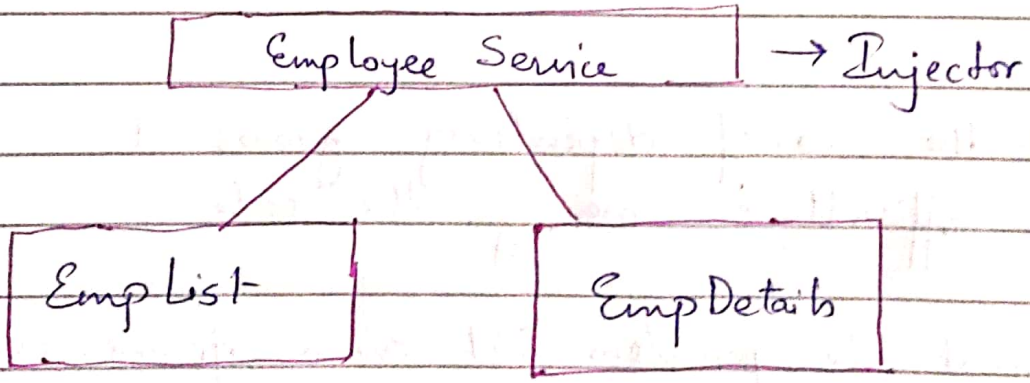
③ DI as a framework :

DI has something called an "Injector" when you register all your dependencies.

So basically Injector is a container which contains all the dependency.

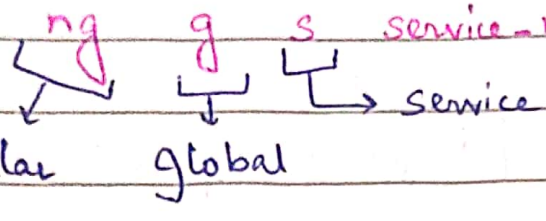
So when need a car (or object), the injector will manage all the dependencies and provide a car.

Ex → ① Define a Employee Service class
② Register with Injector
③ Declare as dependency in EmpList and EmpDetails.



① Define a service class

Command for creating a service:



The above command will create a new file in app/module called

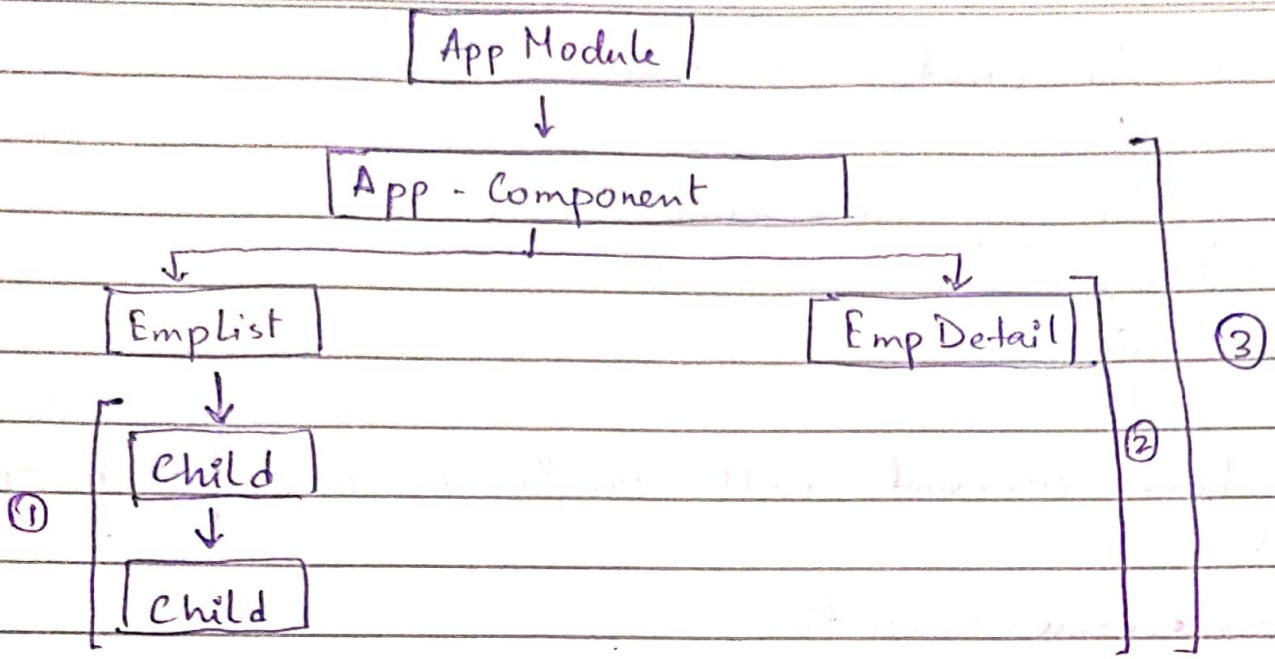
service.name.service.ts

Next step is to register the service with angular injector. If we donot register then service will be just another regular class ~~as~~ according to angular.

Now we need to register in ~~proper~~ proper place coz angular follows Hierarchical DI.

PTO

Register with an Injector



(1) So, if we register the service in emplist then the service is available to only in emplist and its children.

(2) Similarly if we register the servica to app-component then that service is only available in that component block or scope.

(3) So we need to define the servica in app-module is the head of all components. So that any component or child component can use the servica.

[to register a service we use provider metadata]

In app.module.ts :

providers : [service_name],

Dependency for the service :

In any component ts :

constructor (private name : service_name) { }

name of local variable.

type

ngOnInit () {

calling the method that is present in service.

this. employees = this. name. method ()

local variable

variable created in constructor

We are making use of service instance and fetch the service data.

Now local variable employee has the data that present or returned by the method which is defined in service.

We have to do this for all the components that require the service.

~~What~~ Why @Injectable decorator is required?

Injectable decorator tells angular that this service might have injected dependencies.

So, if (in future) wants to inject one service to another service "injectable" decorator is must.

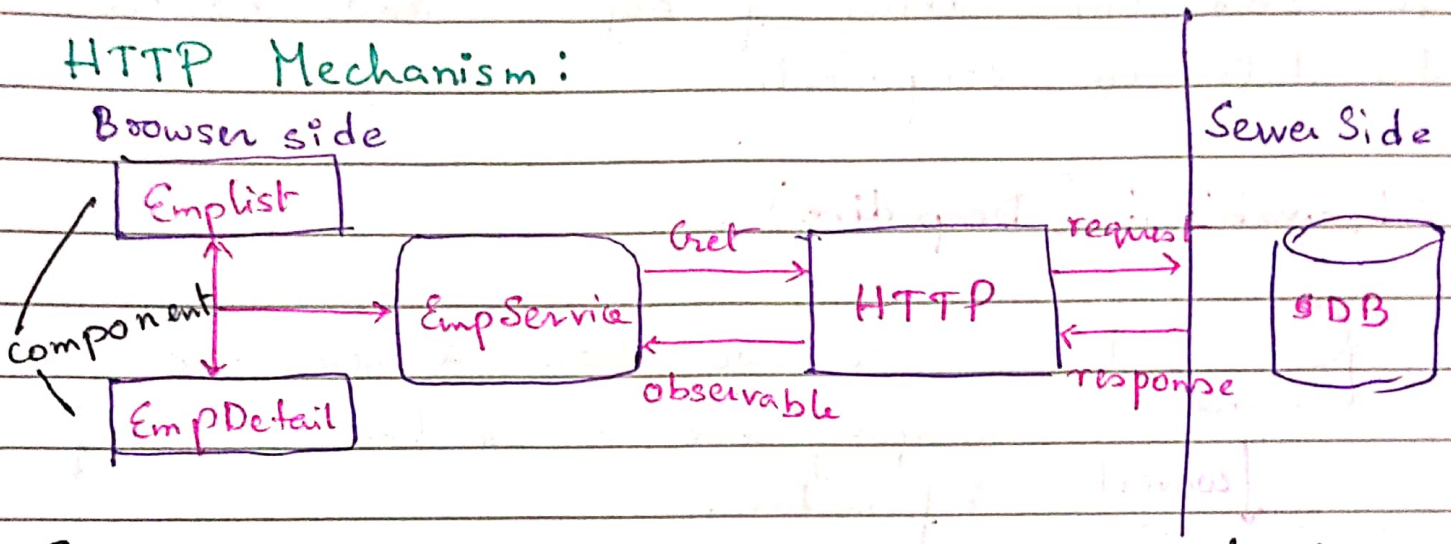
~~So~~ Next question comes we do not have injectable decorator at component but we are using service dependencies.

Ans: @Component decorator lets the component know that it might have dependencies and might make use of dependency injection.

HTTP and Observable (Deep Dive)

In real world we need to fetch data from server and store or import it to services so that all the components are accessable to it.

HTTP Mechanism:



So in previous note we saw we are feteching data only from empservice which is hard coded but in real world projects we need to fetch data from data base or server side.

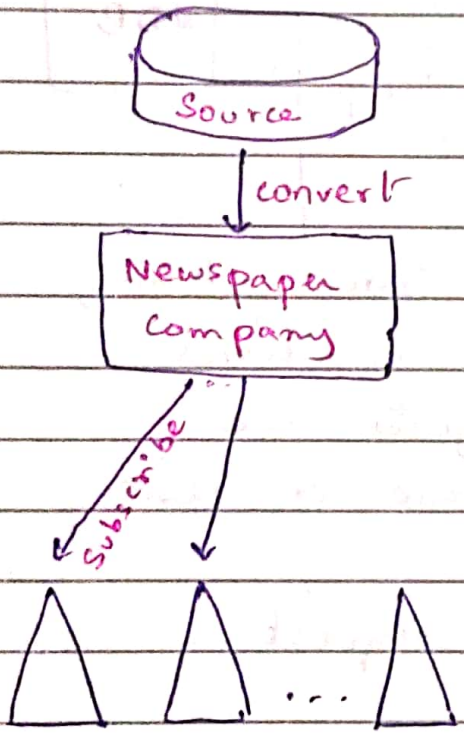
And for that we make "HTTP" request

HTTP "Get" request will hit a web api or a web service which fetch the data from data base and send it back to the call as response with the data

from the server. The response we get back from the HTTP is called an observable.

Now Service needs to cast the returned observable into array of objects and return the data to components.

Observable : (Deep dive)



So we have a newspaper company and we ~~~~ have source of information.

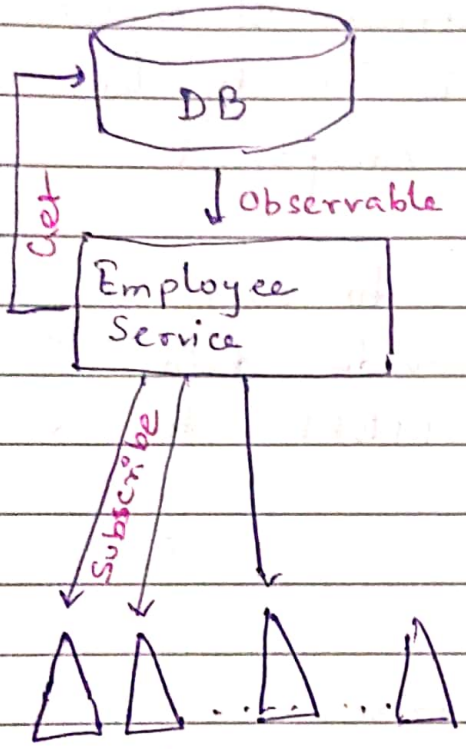
So, company makes a request to the soure for every day news.

As a response to the company's request, the source sends in a sequence of information through out the day.

Now the data is converted into news paper format by the company and now its ready to be distributed among the homes.

But the news paper company only distributes the news paper to the house that are registered or ~~substi~~ subscribed to the company.

Now we will related this with a Angular



source = database or web API or web service

Now, employee service makes request to the database with the "HTTP Get" call. As a response we get back an observable. (*)

What is observable?

- A sequence of items that arrive asynchronously over a period of time.

To understand the above sentence ~~or~~ lets see

HTTP Call → single item
HTTP response ←

therefore an observable is nothing but the HTTP response that arrives asynchronously.

(*) So from server we get a HTTP response as an observable but the format we have recived is not format we can use in our application.
So, once we receive the observable we need to convert the observable into array of objects.
After the conversion the data is ready to be provided to the application component.

But do we provide that data to every single component. "NO". We only provide data the component which is "SUBSCRIBED" to the sevice.

Conclusion:
1. HTTP Get request from service.
2. Receive the observable and cast it into an array of objects.
3. Subscribe to the observable from component
4. Assign array to local variable.

We will see the implementation in Next page

Step 1 : Send get request

First we need to import "HTTP Client Module"

app.module.ts :

```
import : [
        BrowserModule,
        HttpClientModule ],
```

service-name . service. ts :

```
constructor ( private http: HttpClient) { }
```

So, now we have a local variable of Http which will refer to the instance of HttpClient.

Get request :

```
method_name () {
  return this. http. get (" server link here ");
}
```

Optional : →

Instead of putting the URL directly we can do it in another way :

```
private _url : String = " link " ;

getFunction ( ) {
    return this. http. get ( this._url ) ;
}
```

Step 2 : Cast the observable which we receive as response to employee array :

First we need to ~~see~~ create interface :

employee. ts :

```
export interface IEmployee {
        id : number,
        name : string,
        age : number
    }
```

=>] → this assigns the data received from the observable to this the local property.

We now have employee type that the observable can be cast into.

employee.service.ts :

```
.
.
.
getEmployee() {
                ↳ : observable<IEmployee[]>
  return this.http.get<IEmployee[]>(this.url);
}
```

3. Subscribe to the observable :

```
public employees = [];
constructor(private _employeeService : EmployeeService) { }

ngOnInit() {
// this.employee = this._employeeService.getEmployee()

this._employeeService.getEmployee().subscribe( data =>
this.employees = data);
```

this.employees → argument of the function

data → body of the function

So, we are assigning data to the property.

① We have a instance of the employee service
② we use this to call get ~~emp~~ getEmployee method.
③ This method (that getEmployee) returns an observable
④ So to receive data we need to subscribe to it.
⑤ Once we subscribe ~~it~~ to the observable the employee data arrives asynchronously.
⑥ We assign the data to the class property using fat arrow (=>) syntax.

Component Deepdive:

A component is made of 3 parts:

1. Component Template
2. Class
3. Meta data

① Template: this represents the view. This is created using HTML and it will be the user interface for your application.

② Class: Class is nothing but code that supports the view. and this is created using type script. It also contains data properties and methods that can be used to control the logic of the view.

③ Meta Data: This is the information that angular needs to decide if the particular class is in fact an angular component or a class. Meta data is defined using a decorator which is a feature in type script. A decorator is just a function that provides information about the class attached to it. and for component we use component decorator.

```
@Component({
    selector: 'app-root',
    templateURL: './app.component.html',
    styleURLs: ['./app.component.css']
})
```

the above is the meta data attached in a form of decorator and to be more specific "the component" decorator.

Component decorator is basically a function that attaches to the class right below it.
This decorator tells angular, hey this is not a plain class this is a component.
It contains both meta data and the template which represents the view.
So as part of metadata we have

1. Selector
2. template Urls
3. Style Urls

① Selector: It is basically a custom HTML tag that can be used to represent this component. When you specify a selector in HTML, angular renders the component's template in its place.

② templateUrl: This points to HTML file, that represents the view for the component.

③ styleUrls: This is css sheets which apply only to the component.

To create new component:

ng g c test] → component name

ng → angular
g → global
c → component

After this we can see 4 new files are created and also there will be one update in the app module.

NOTE: By default in Angular while creating a component it generates 4 files
① HTHL ② CSS ③ TS ④ specs (for testing)

NOTE: If you want to create a component without specs

```
ng g c test --spec
```

Inside app module there is an update.
So, as soon as an component is created, angular updates the app module as this component is a part of the application.

inside app.module.ts

```
@NgModule({
  declaration:[
    AppComponent,
    TestComponent
  ],
```

→ all the components of an angular application is showed here.

this is done by angular CLI

NOTE: Angular application will have one root component which is the app-component and all the other component will fall under the app-component.

Let's see what are the changes we can make to a component:

① Selector: There are 3 ways to specify a selector

① We specify the selector <app-test> which we use it as custom HTML tags

② We can also use it as a class. to do that begin your selector with . and we can use this selector as a class name.

③ Third way is to enclose the selector using square brackets. Now we can use this selector as an attribute, to this div tags.

② templateUrl : This point to the file that contains the HTML file. It is also possible to specify the template in line. (in the same ts file) to do that

```
template : '<div>Inline template</div>',
```

③ styleURLs : This points to the css files. We can also have inline css.

NOTE: One point to remember angular component only uses one HTML template URL but it can use multiple css urls.

So, in component metadata we see only one templateUrl but in styleUrls it ~~is~~ declared ~~as~~ as an array of url.

Interpolation (Deep Dive)

Say we create a component which has only one file that is type script

```
@component({
    selector: 'app-test',
    template:
        <h2>
        Welcome to BJS
        </h2>,
    Styles: []
})
```

this is static template which will only print "Welcome to BJS".

But we wants to make the name "BJS" as dynamic that it will show the name that the user logged into. which might come from web services or web api.

So for that what we do is create new property in the class, lets call it name and assign a name to "BJS".

```
export class Testcomponent implements OnInit {

    public name = "BJS";
                    ↳ if we change from BJS → South

    constructor() { }

    ngOnInit() { }

    greetUser() { return "Hello " + this.name; }
}
```

Now the question is how do we get this value inside template? The answer is {{ }} and inside the curly braces we mention the property name that we have declared in the class.

The syntax of a property or an expression within double curly braces is known as interpolation in angular.
Using interpolation we are asking angular to evaluate the content inside the curly braces and display the value when the component is rendered in the browser.

```
@Component({
  selector: 'app-test',
  template:
    <h2>
      Welcome {{name}}
    </h2>,
  styles: []
```

→ this the name property will be dynamically changed to "Sourik".

This is a simpliest way of binding data from class to the template.

Now lets see what interpolation can and can not do!

① We can perform arithmatic operation.
Ex → `<h2> {{2+2}} </h2>`

② We can also perform concatination.
Ex → `<h2> {{"Welcome " + name}} </h2>`

③ We can also use JS properties and methods.
Ex → `<h2> {{name.length}} </h2>`
`<h2> {{name.toUpperCase()}} </h2>`

④ We can also call methods defined in component.
Ex → `<h2> {{greetUser()}} </h2>`

(*)

This can't do in interpolation:

① Assigning the expression to a variable

Ex: `<h2> {{ a = 2+2 }} </h2>`

② Access to bcaw global variables like window, screen and so on.

Ex: `<h2> {{ window.location.href }} </h2>`

We can do the above by binding the value to a property.

Property Binding:

- Before starting we do need to understand the difference between HTML attribute and DOM property!

① Attribute and property are not same.

② Attribute is defined by HTML element. Model)
Properties is defined by DOM (Document Object

③ Attribute initialize DOM properties and then they are done. Attributes value never change once the initialization is over.
Property values can change.

```
@Component({
  selector: 'app-test',
  template:
    <h2>
      Welcome {{name}}
    </h2>
    <input [id]="myId" type="text" value="BJS">
    <input id="{{myId}}" type="text" value="BJS">
  styles: []
})
export class Testcomponent implements OnInit {
  public myId = "testId";
  constructor() { }
  ngOnInit() { }
}
```

Why do we need property binding if we can use interpolation in property binding?

Well, there is a limitation to interpolation that is it can only work with string values and there are HTML properties that are boolean properties that we may need sometimes to bind to.

Lets see an example:

consider an disabled attribute of an input element as be by default it is always set to false.

```
<input disabled = "false" id = "{{ myId }}" type = text
    value = "Vishwas" >
```

By default disabled is true so even if the value we set to false still the input attribute will be disabled = "true".

So, the above is achieved through property binding.

```
<input [disabled ] = "false" id = "{{ myId }}" type = text
    value = "Vishwas" >
```

Now the input element is now ~~de~~ enabled.