**Angular:**

**How angular application works ? What happens when appplication starts ? Where is the stating point ?**

angular.json -> main.ts -> app.module.ts -> index.html

The **angular**.**json file** at the root level of an **Angular** workspace provides workspace-wide and project-specific configuration defaults.

A screen shot of a computer program

AI-generated content may be incorrect.

**main**.**ts file** is the entry point of our web-app. It compiles the web-app and bootstraps the AppModule to run in the browser.

A screen shot of a computer program

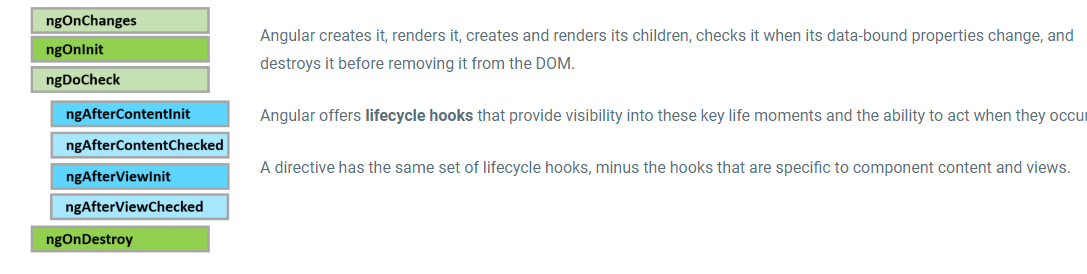
AI-generated content may be incorrect.

In Angular, the app.module.ts is the root module of the Angular application. It's where you define the structure, components, services, and other features of your application.

A screen shot of a computer program

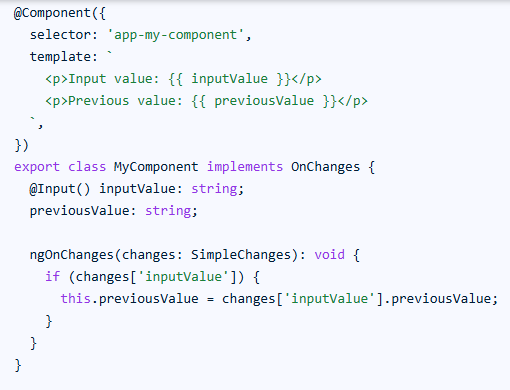
AI-generated content may be incorrect.

**LifeCyle of Angular application ?**



**Explain the ngOnChanges() hook and give an example of its use.**

The ngOnChanges() hook is a life cycle hook that is called when any data-bound input property changes. It receives an object containing the current and previous property values as arguments. This hook is useful for reacting to input property changes, such as updating the component's state or validating input values.



A screenshot of a computer

AI-generated content may be incorrect.

### Describe the ngOnDestroy() hook and its importance in the Angular life cycle.

A: The ngOnDestroy() hook is a life cycle hook called just before Angular destroys the component or directive. It is important for cleaning up resources, such as subscriptions or event listeners, to prevent memory leaks and ensure proper application performance.

A screenshot of a computer

AI-generated content may be incorrect.

**Explain the concept of dependency injection**

**What is the purpose of the async pipe?**

****

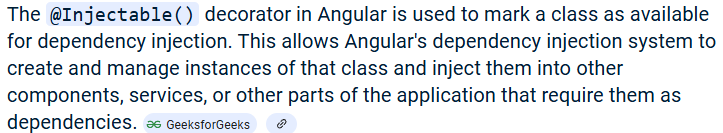
**What are pipes ?Why are we using ?**

**What is pure & impure pipes ?**

**Different methos to pass** values **from one component to another ?**

1. [Parent to Child: via Input](https://www.samarpaninfotech.com/blog/methods-to-share-data-between-angular-components/#h-method-1-parent-to-child-via-input-decorator)
2. [Child to Parent: via Output() and EventEmitter](https://www.samarpaninfotech.com/blog/methods-to-share-data-between-angular-components/#h-method-2-child-to-parent-via-output-decorator-and-eventemitter)
3. [Child to Parent: via ViewChild](https://www.samarpaninfotech.com/blog/methods-to-share-data-between-angular-components/#h-method-3-child-to-parent-via-viewchild-decorator)
4. [Unrelated Components: via a Service](https://www.samarpaninfotech.com/blog/methods-to-share-data-between-angular-components/#h-method-4-unrelated-components-via-a-service)

**Subjects & Observables. Difference & use**

****

**Session vs Cookie**

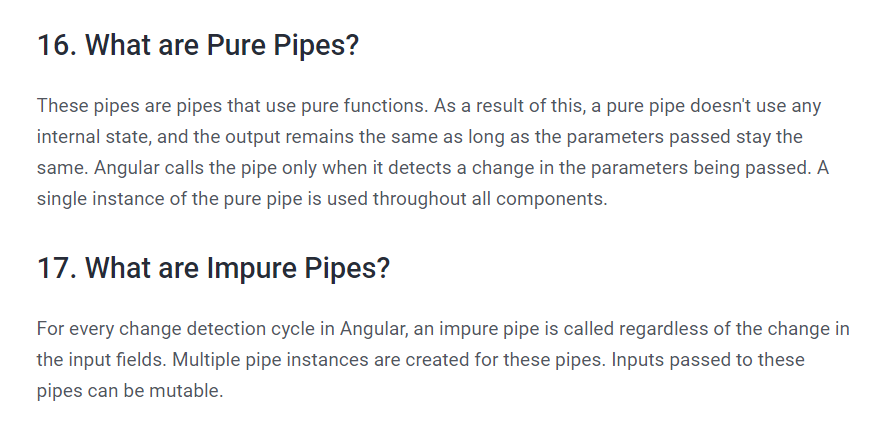
**A screenshot of a chat

AI-generated content may be incorrect.**

### What are decorators in Angular?

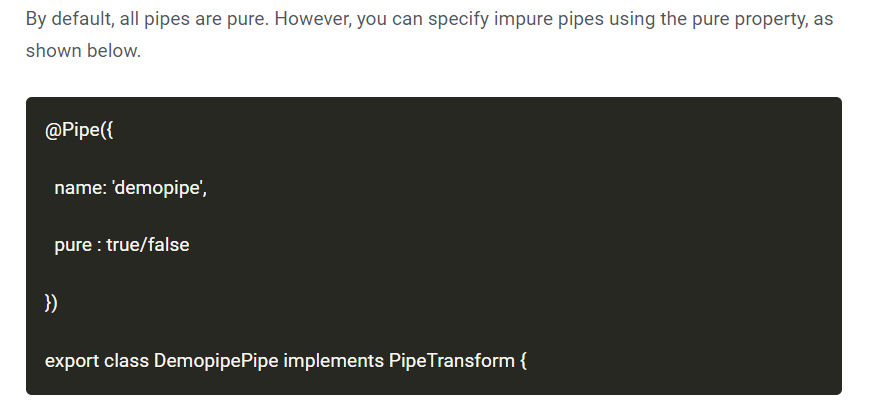
Decorators are a design pattern or functions that define how Angular features work. They are used to make prior modifications to a class, service, or filter. Angular supports four types of decorators, they are:

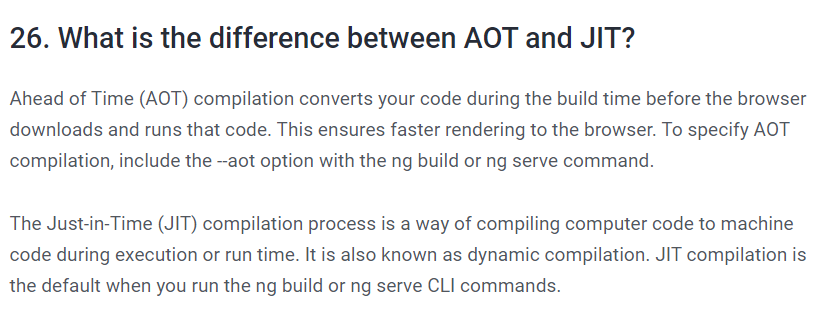
1. Class Decorators - @Component and @NgModule
2. Property Decorators -  @Input (), @Output, @ReadOnly (), @Override ()
3. Method Decorators - @HostListener
4. Parameter Decorators -  @Inject ()

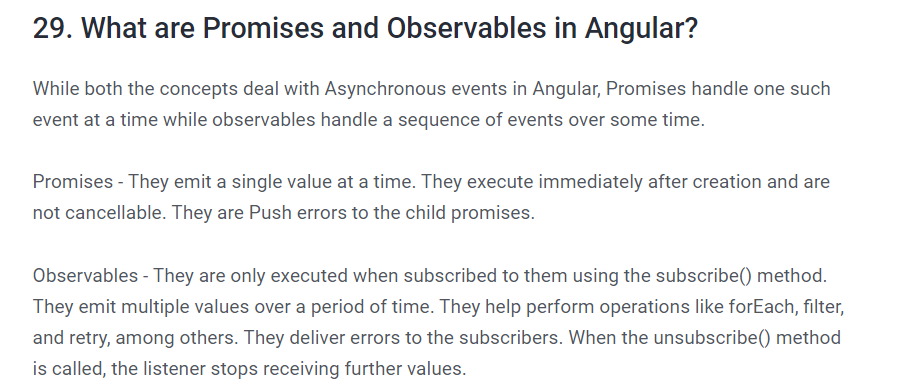


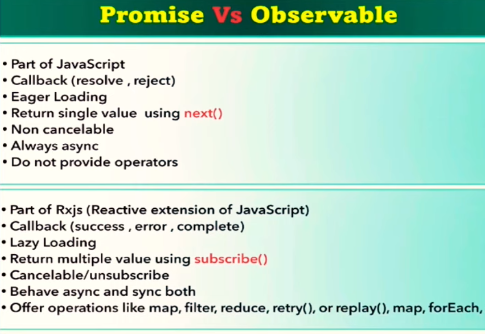
 pure pipes are executed when a change is made to a primitive input value, while impure pipes are executed during every change detection cycle

 Pure pipes are only executed when the input value changes, while impure pipes are executed on every change detection cycle, regardless of whether the input value has changed





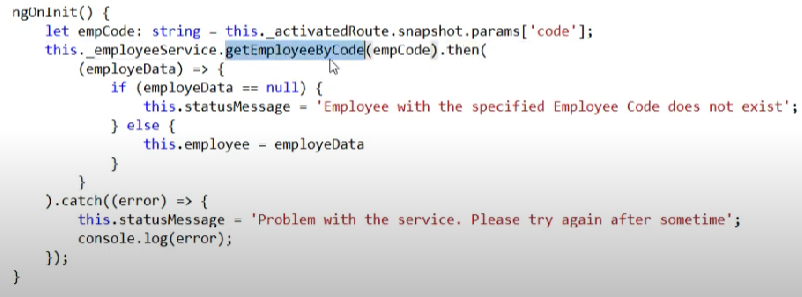




**Promise**

* Emits a single value
* Eager Loading – API call is executed even if the promise is not called by then(). (check in Network tab)

Eg: GetEmployeeById() – returns only single data object

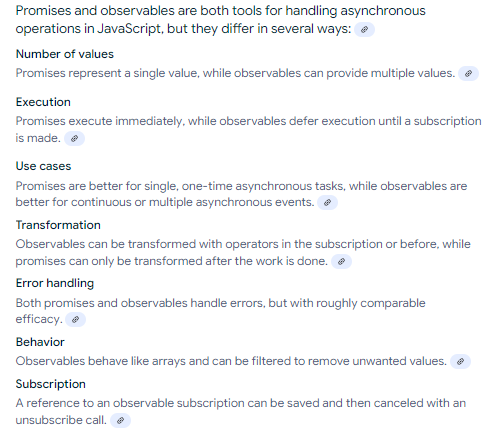


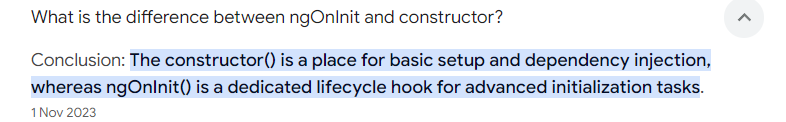
**Observables**

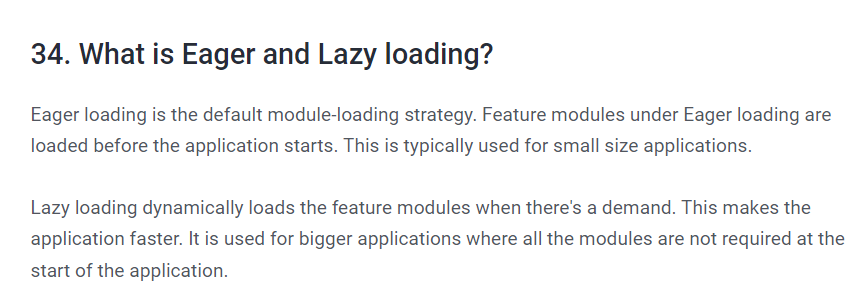
* Emits multiple values over a period of time
* Lazy loading – API call will be called only if the observable is subscribed. (check in Network tab)

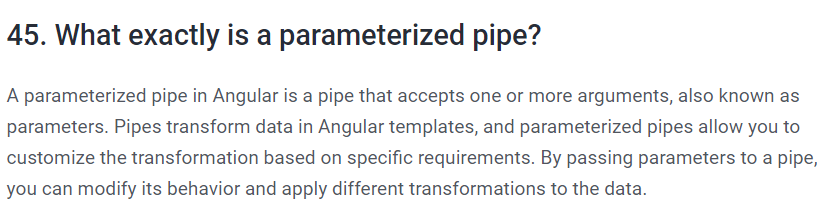
Eg : GetAllEmployees() – returns multiple data

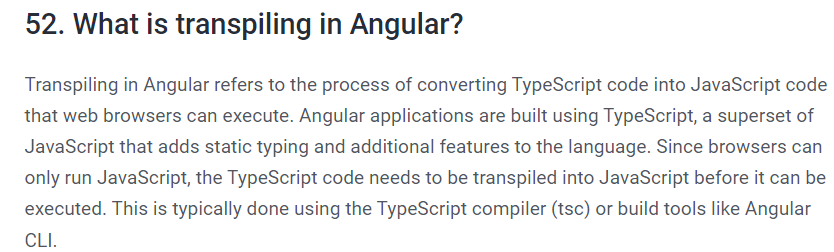


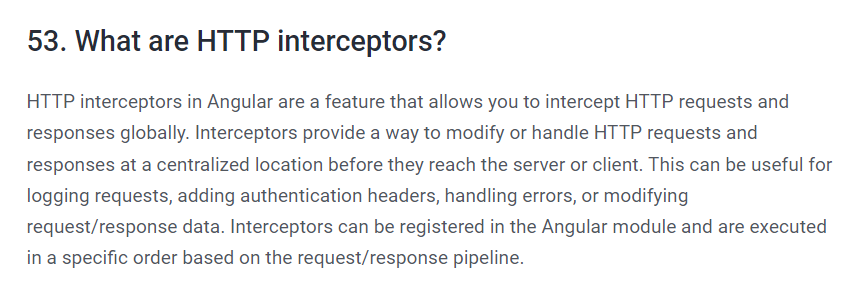


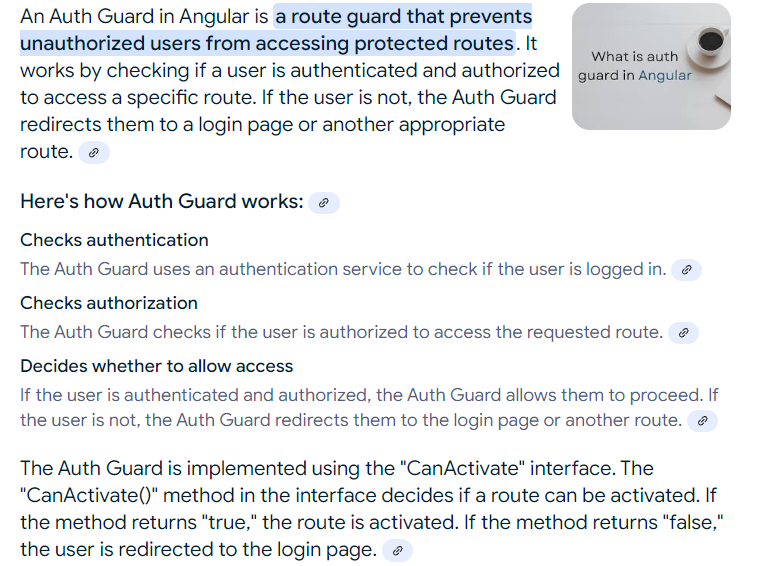


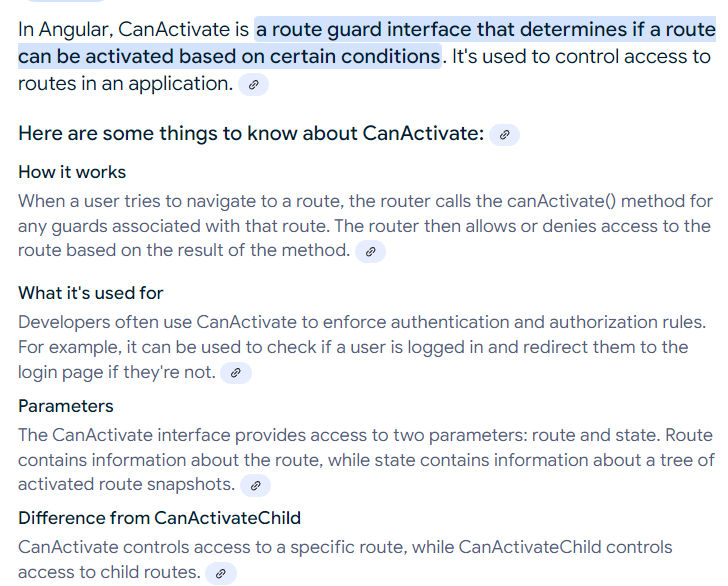


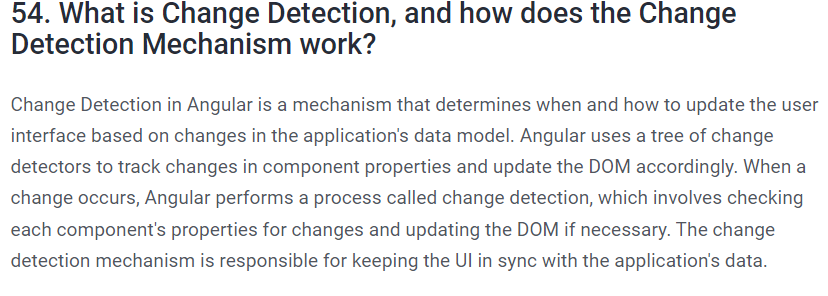


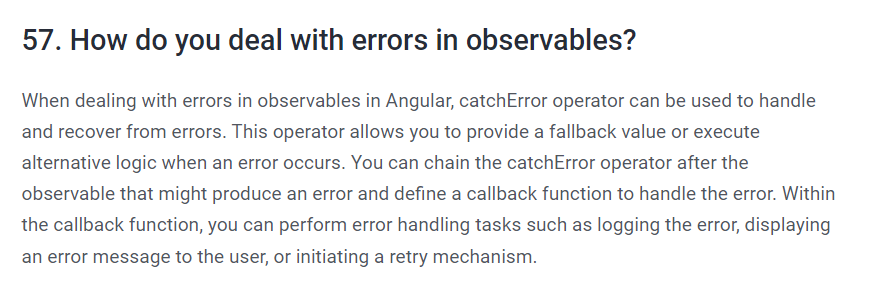


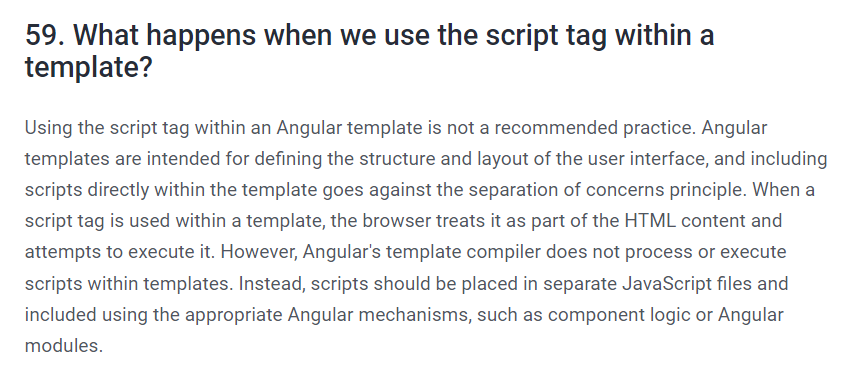












**What is the need of angular ?**

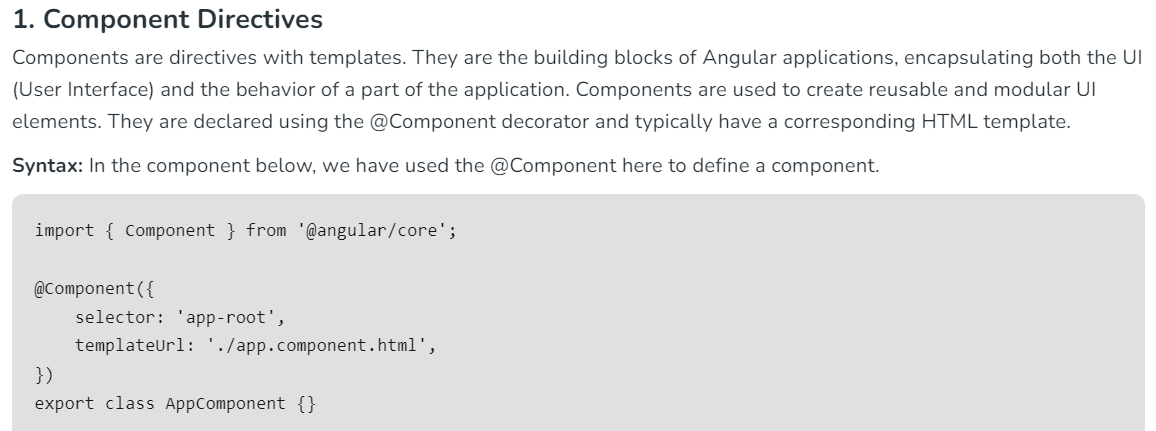
Angular is a full-featured JavaScript framework that provides an extensive library of tools to support your development process

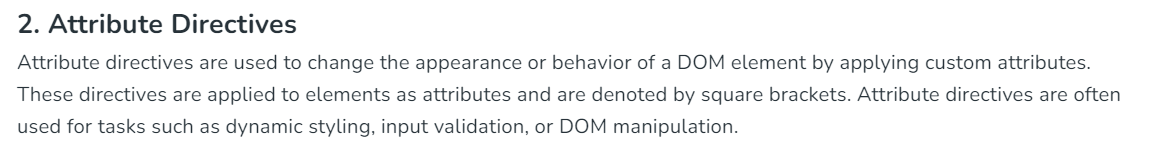
Angular features like two-way data binding, data-driven templates, dependency injection, and testing utilities help improve the quality of your code while simplifying development and maintenance processes.

**3 types of directives ?**

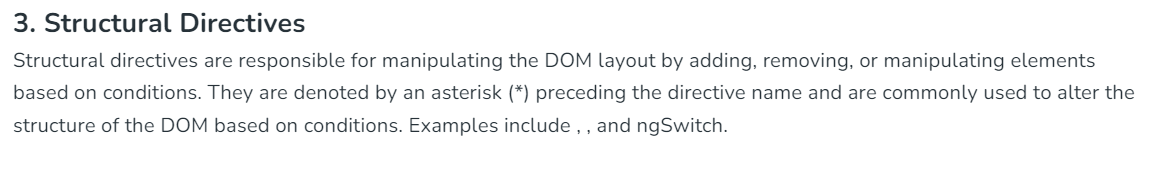
**Directives** are markers in the Document Object Model(DOM). Directives can be used with any controller or HTML tag which will tell the compiler what exact operation or behavior is expected

* [1. Component Directives](https://www.geeksforgeeks.org/built-in-directives-in-angular/#1-component-directives)
* [2. Attribute Directives](https://www.geeksforgeeks.org/built-in-directives-in-angular/#2-attribute-directives)
* [3. Structural Directives](https://www.geeksforgeeks.org/built-in-directives-in-angular/#3-structural-directives)



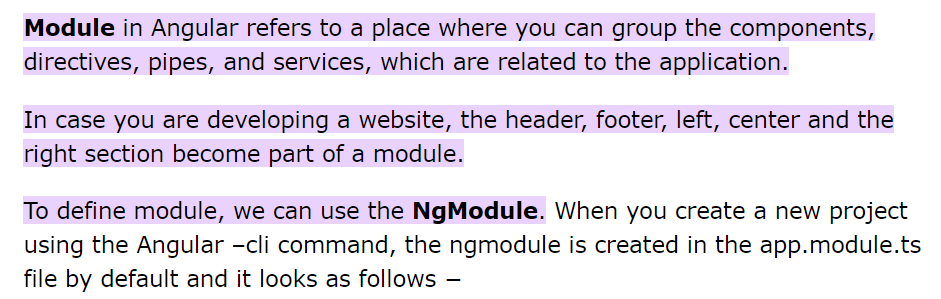


**Eg: ngClass, ngStyle, ngModel.**

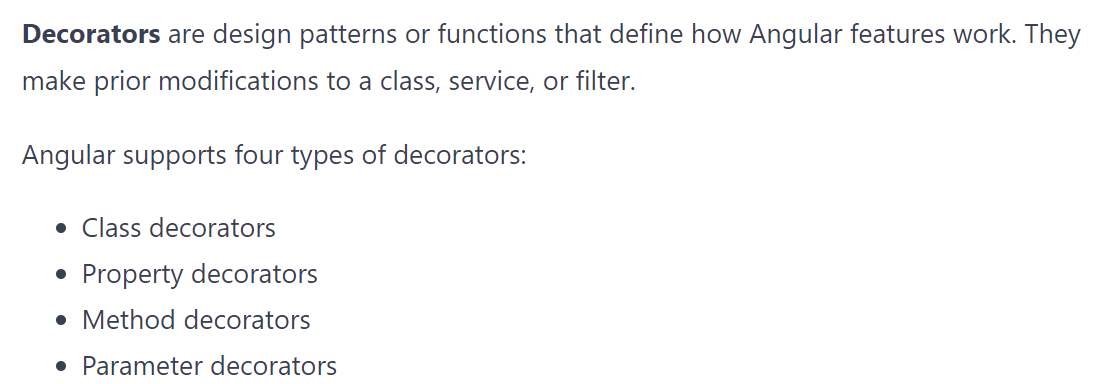


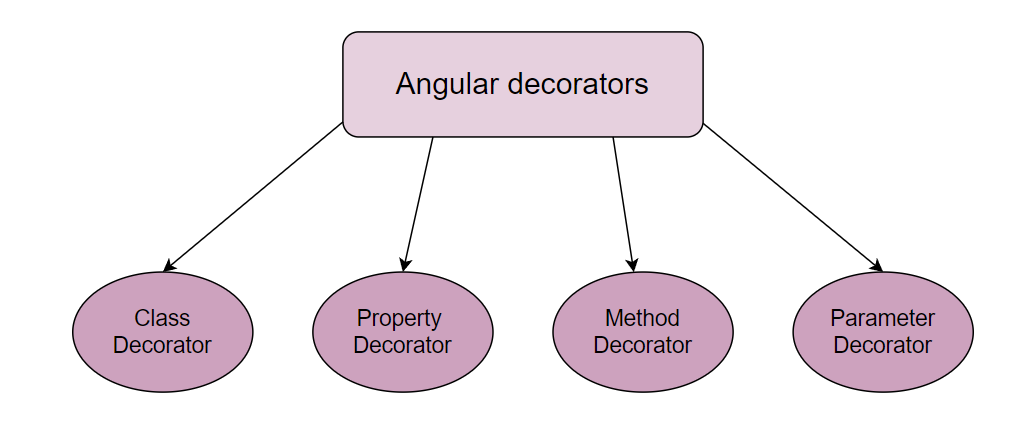
**Eg: ngIf, ngFor, ngSwitch**

**Module**



**Decorator**





### Class decorators

A **class decorator** tells Angular if a particular class is a component or a module.

There are various class decorators in Angular, and among them, @Component and @NgModule are widely used.

### Property decorators

**Property decorators** allow us to decorate some properties within our classes. We can quickly identify why we use any particular property of a class like @Input(), @Output(), @Override(), and so on. We can place the @Input() decorator above the property with this decorator. The AngularJS compiler will create an input binding with the property name and link them.

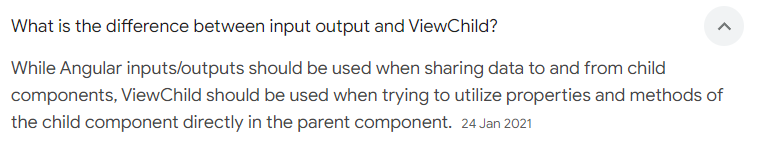
### Method decorators

**Method decorators** are used to decorate the method defined inside our class with functionality. A typical example of a method decorator is @HostListener.

### Parameter decorators

**Parameter decorators** are applied to the constructor parameter of the class and are used when we need to tell Angular to inject a particular provider in a constructor.

@Inject() is a widely used parameter decorator. We use this decorator to inject services in Angular classes.



**Annotations**

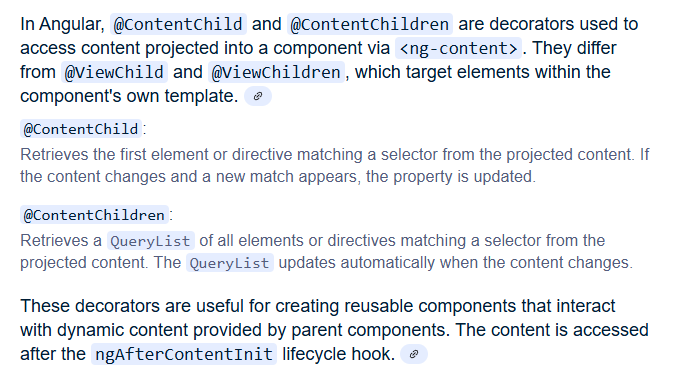
In Angular, annotations are a way to add metadata to a class or a class member using special decorators. These decorators provide additional information to Angular about how a class or its members should be treated or used.

Eg: @component, @NgModule etc..

**Lazy loading , Services , Dependency Injection**

**Component life cycle.**

**ContentChild & ContentChildren**

****

**ViewChild & ViewChildren**

**A blue and white text on a dark background

AI-generated content may be incorrect.**

**A close-up of a text

AI-generated content may be incorrect.**

**A text on a white background

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Content Projection**

**RxJs, Observable and Observer.**

**Template and reactive forms validations**

**Angular pipes , Change Detection & Making HTTP calls.**

**Ng-content vs ng-container**

**ngModule vs Component**

**ngFor vs ngRepeat**

**A close-up of a white background

AI-generated content may be incorrect.**

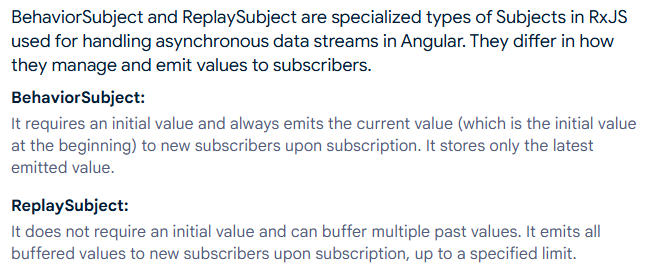
**A screenshot of a computer

AI-generated content may be incorrect.**

**Pipers by Filters**

**Angular Material vs Bootstrap**

**Behavioral subject vs ReplaySubject**

****

**A white background with black text

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.** **A computer screen shot of a program

AI-generated content may be incorrect.**

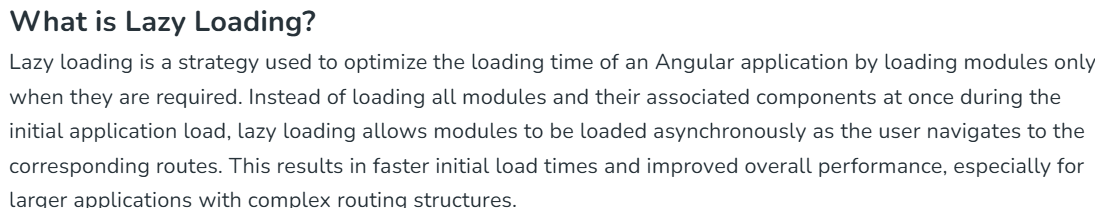
**Routing in Angular**

**Observables vs Promise**

**Pure vs Impure pipes**

**To create custom pipe**

**Implement lazy loading in Angular**

****

**Eg: Normal Lazy loading**

****

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer code

AI-generated content may be incorrect.**

**A close-up of a question

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer code

AI-generated content may be incorrect.**

**Authentication in Angular (Auth guards)**

**A screen shot of a computer

AI-generated content may be incorrect.**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Add UserService in AuthenticationGuard to validate the user:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Role based Authorization**

Load the user details from http call which contains roles data

**A screen shot of a computer screen

AI-generated content may be incorrect.**

**New guard for Role: HasRoleGuard**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A computer screen with text

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**