TOP 200+

Angular

INTERVIEW QUESTIONS

ROHATASH KUMAR

ABOUT THE BOOK

This book contains **Top 250+ Microsoft Azure** interview questions designed to help you prepare effectively for Azure and Cloud technical interviews.

The questions are carefully selected to cover **Azure fundamentals**, core services, real-time scenarios, an **advanced** concepts that are most commonly asked in real interviews.

Whether you are a **beginner**, developer, or **experienced** professional, this book will help you **strengthen your Azure knowledge, build confidence**, and succeed in interviews.

ABOUT THE AUTHOR

Rohatash Kumar has over **15 years of experience** in software development. He has helped many candidates succeed in technical interviews at well-known tech companies by sharing **his knowledge and guidance.**

Angular Chapters

1. Angular-Sample Questions

Q. What is Angular? What are Angular Advantages?

Q. What is RxJS? What are the types of operations in RxJS?

Q. What is Observable? How to implement Observable?

Q. What is the difference between JIT and AOT in Angular?

Q. What is String Interpolation in Angular?

Q. What is Decorator?

Q. What are Pipes? What are the types of Pipes?

Q. What is the difference between Angular Pure Pipe and Impure Pipe?

Q. What are Arrow Functions in Typescript?

Q. How to use Dependency Injector with Services in Angular?

Q. What is Angular? What are Angular Advantages?

Angular is a TypeScript-based front-end web application framework developed and maintained by Google. It is used to build fast, scalable, and maintainable single-page applications (SPAs).

In simple words

Angular helps you build dynamic websites and web apps where the page does not reload again and again-only the required data updates.

Where Angular is used

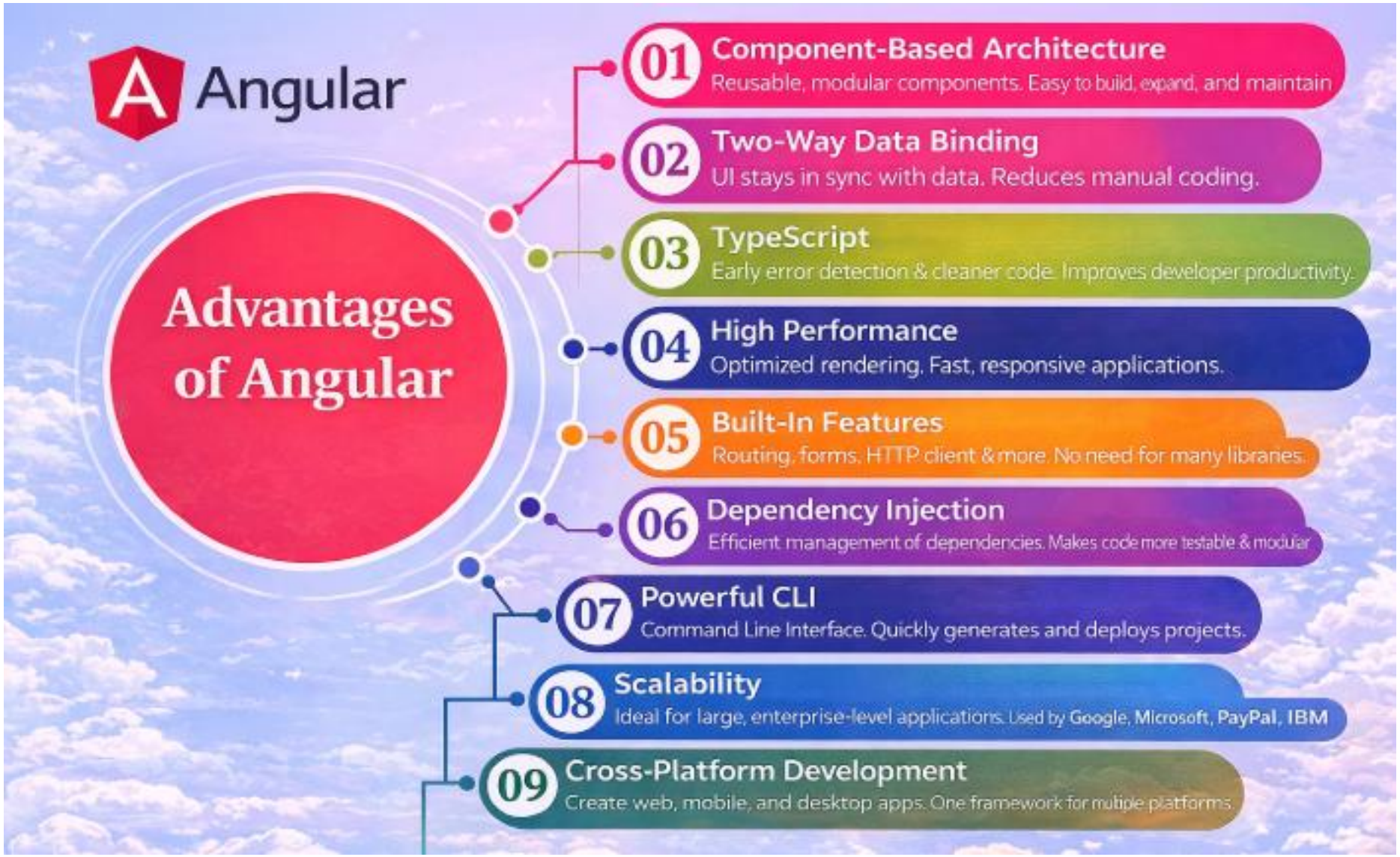
- Enterprise-level web applications
- Admin dashboards
- Banking & finance apps

Simple Example

Think of Angular like a **smart TV interface**.

- Screen doesn't reload completely
- Only required sections update
- Smooth navigation

Q. What are Angular Advantages ?



Q. What is RxJS? What are the types of operations in RxJS?

RxJS (Reactive Extensions for JavaScript) is a library used in Angular for handling **asynchronous + event-based programming** using **Observables**.

In simple words

👉 RxJS helps you handle async data streams easily (API call, mouse click, search typing, websocket, timer, etc.)

Why RxJS is used?

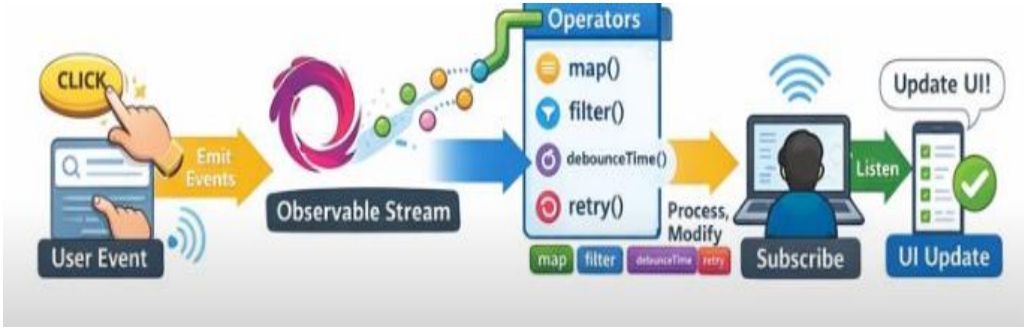
Because in real applications, data is not always one-time.

Examples of continuous data:

- User typing in textbox ✅ (every keypress)
- Button clicks ✅
- API requests ✅
- WebSocket live updates ✅
- Timer / interval ✅

RxJS makes these easy to control and manage.

All this is RxJS power ✅



Real-world Example -Train station announcement

Announcements come continuously:

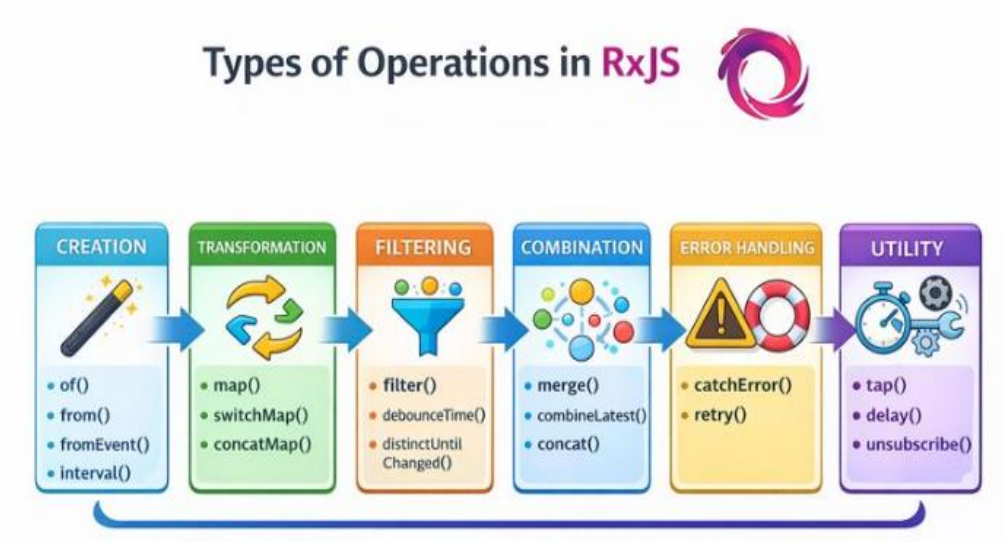
- "Train A arrived"
- "Train B delayed"
- "Train C departed"

This is like an **Observable stream**.

You (listener) can:

- listen (subscribe)
- stop listening (unsubscribe)
- filter announcements
- take only first 3 announcements
- retry if missed

What are the types of operations in RXJS ?



Simple RxJS Example

```
import { of } from 'rxjs';

of(10, 20, 30).subscribe(data => {
  console.log(data);
});
```

Output

```
10
20
30
```

Example with Operator (map)

```
import { of } from 'rxjs';
import { map } from 'rxjs/operators';

of(1, 2, 3)
  .pipe(map(x => x * 10))
  .subscribe(result => console.log(result));
```

Output

```
10
20
30
```

RxJS in Angular (Most important use)

```
this.http.get("https://api.com/users")
  .subscribe(res => {
    console.log(res);
  });
```

Q. What is Observable? How to implement Observable?

An Observable is a data stream that can send data over time.

👉 It can send:

- 1 value
- many values
- after some delay
- continuously (like live updates)

In Angular, HttpClient.get() returns **Observable**.

Real-world Example

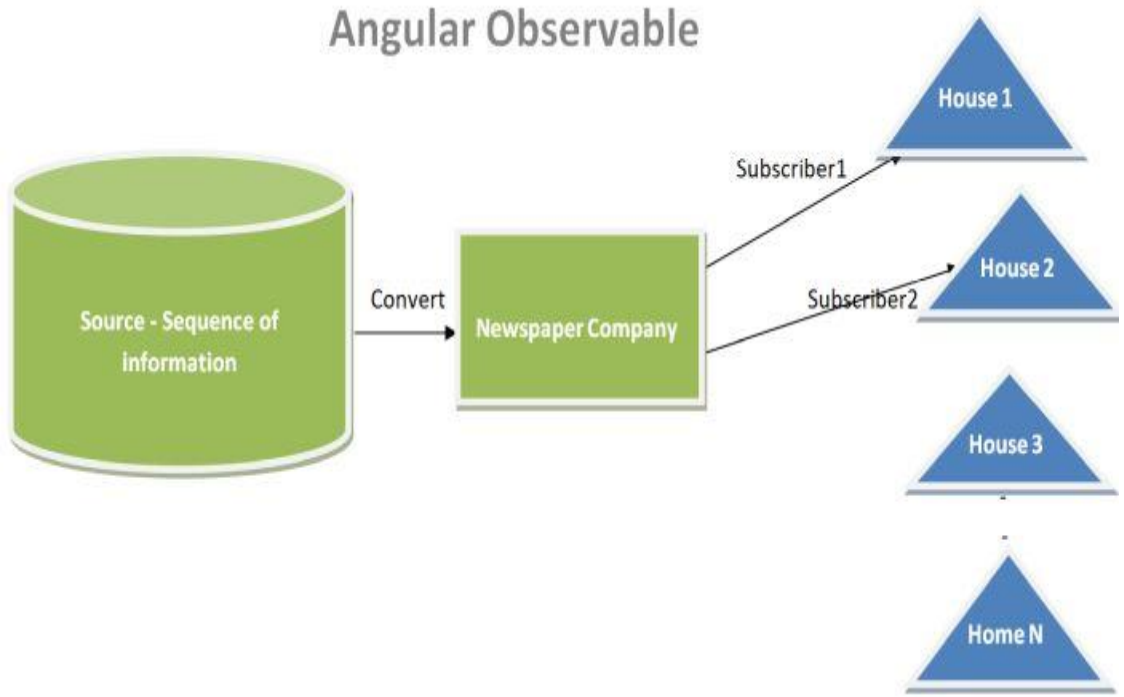
📺 YouTube Live Stream

- Stream keeps sending video/audio continuously.
- You can subscribe (watch)
- You can unsubscribe (stop watching)

That's exactly how Observable works

Example

Here is a real-world example from that we can understand concept of observables in better way. You may think about newspaper where subscribers receive newspaper while non-subscribers do not. The below image show observable concept.



Q. What is the difference between JIT and AOT in Angular?

JIT compiles Angular code inside the browser at runtime.

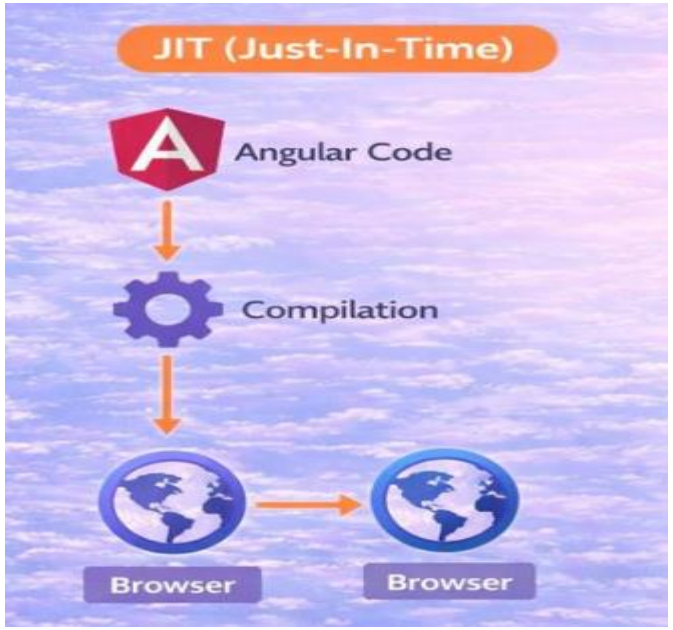
Used when

```
ng serve
```

Simple Example

JIT = Cooking food after customer orders.

How it Works



AOT compiles Angular code **during the build process**, before deployment, resulting in faster performance, smaller bundles, and better security.

Used when

```
ng build --configuration=production
```

Simple Example

AOT = Cooking food in advance and serving immediately.

How it Works

Q. What is String Interpolation in Angular?

String Interpolation

String interpolation is used to bind component data into HTML template using {{ }} and it is one-way binding (Component → View).

Example

app.component.ts

```
export class AppComponent {
  userName = 'Rohatash';
  age = 25;
}
```

app.component.html

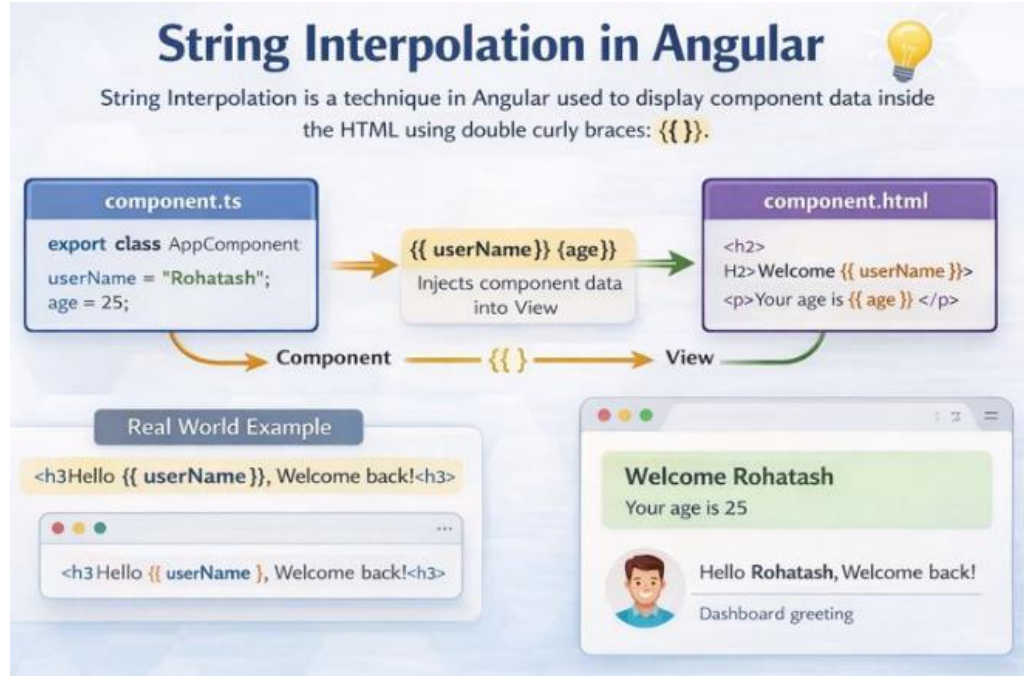
```
<h2>Welcome {{ userName }}</h2>
<p>Your age is {{ age }}</p>
```

Real World Example

Dashboard of an app

If a user logs in, Angular shows their name:

```
<h3>Hello {{ userName }}, Welcome back!</h3>
```



Q. What is Decorator?

A Decorator in Angular/TypeScript is a **special function** (written with @) that **adds metadata (extra information)** to a class, property, method, or parameter.

Example of Decorator

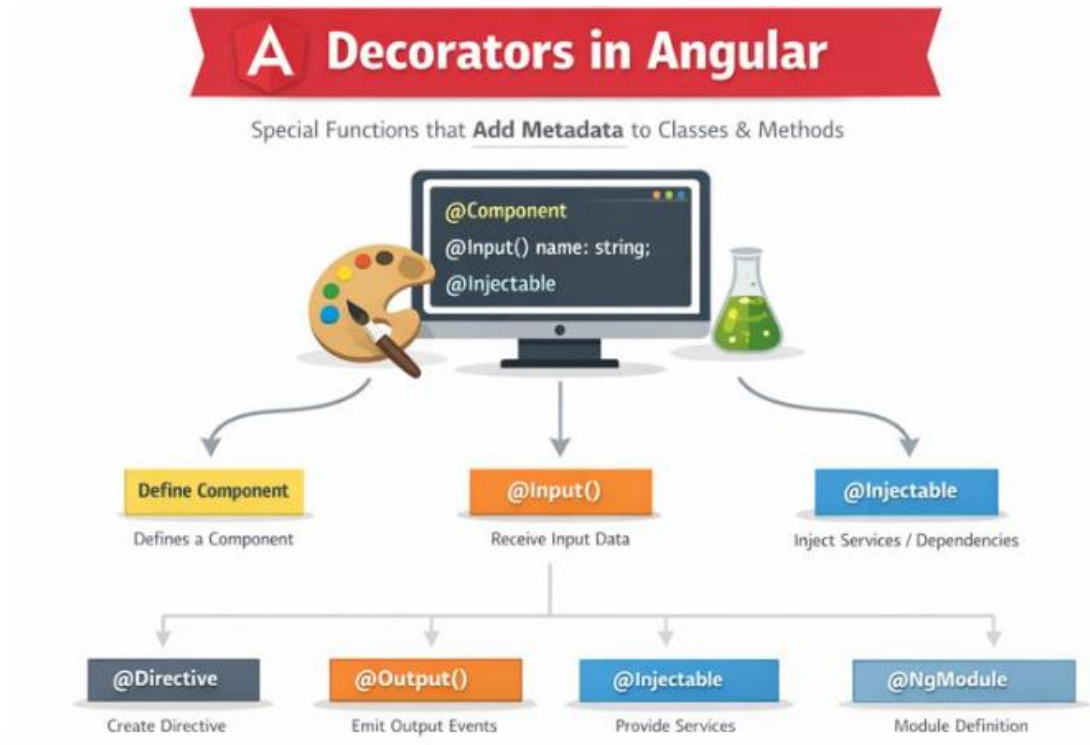
✅ Component Decorator

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
}
```

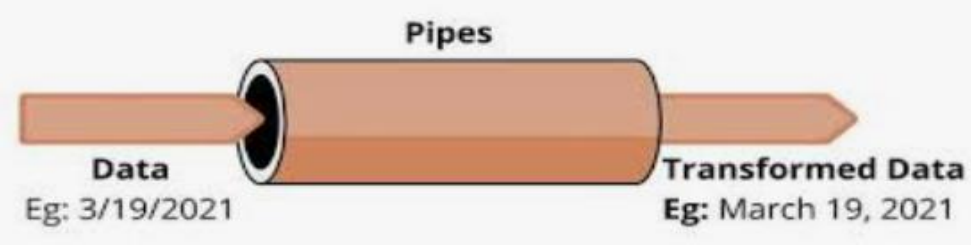
Here, @Component is a **decorator** which tells Angular:

- This class is a component
- Template file is app.component.html
- Selector is app-root



Q. What are Pipes? What are the types of Pipes?

In Angular, Pipes are a simple way to transform data in the HTML template (view) without changing the original value in the component.

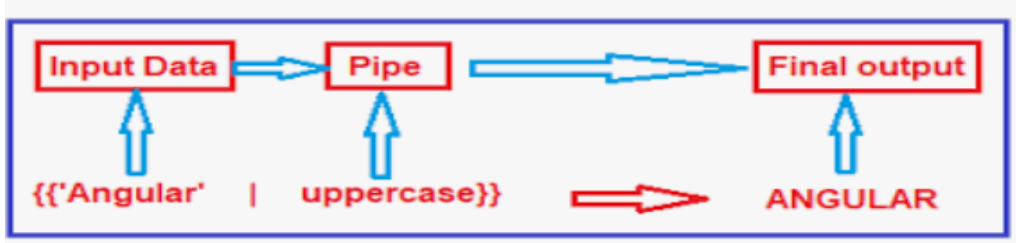


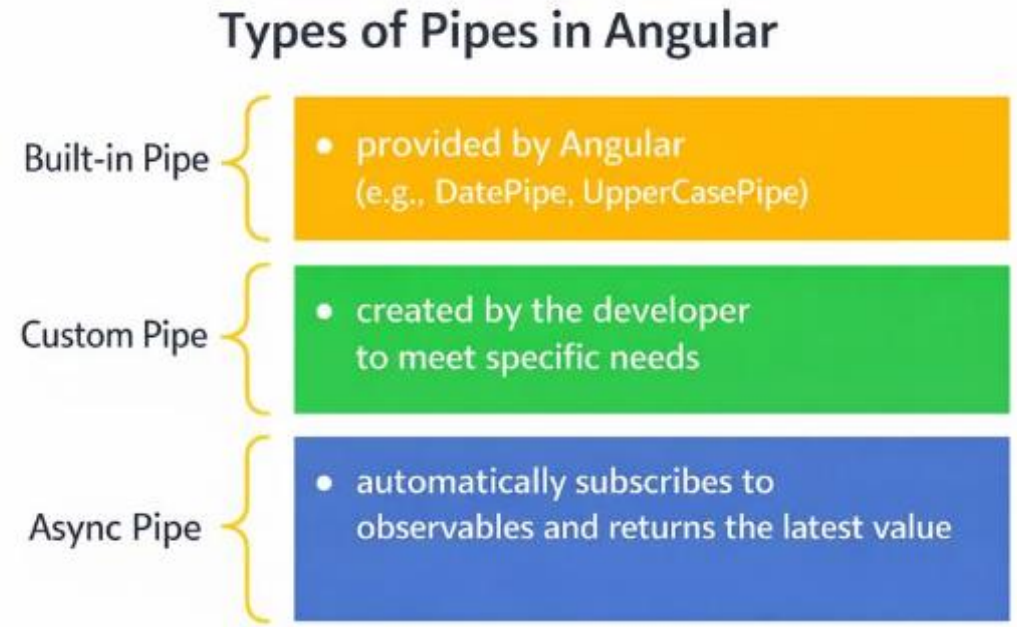
Example

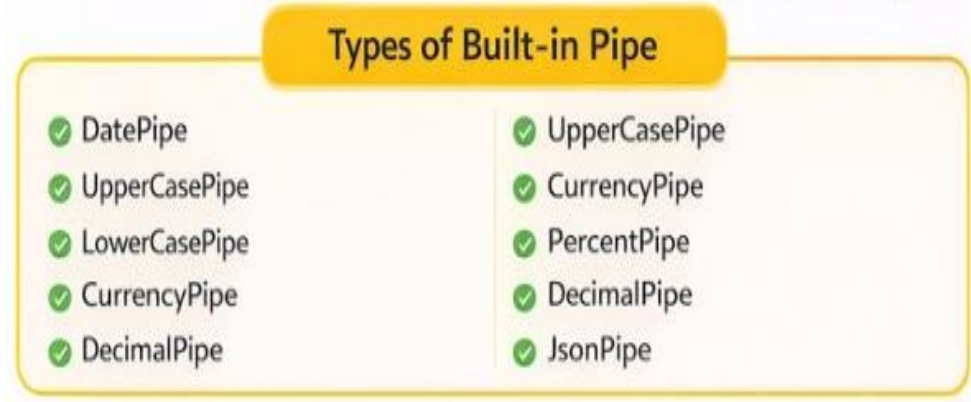
Convert text to uppercase, format date, currency, percent, etc.

{{ 'angular' | uppercase }}

Output - ANGULAR







Q. What is the difference between Angular Pure Pipe and Impure Pipe?

1. Pure Pipe

A Pure Pipe executes only when the input value changes (by reference). It is optimized for performance — Angular does not call it on every change detection cycle.

When it runs

- When the input value or object reference changes.
- When a new array/object/string/number is assigned.

Simple Analogy

Think of “reference” like a house address:

- The array/object is a house.
- The reference is the address.

If you paint the house (change contents), address stays same → Angular doesn’t notice.
If you move to a new house (create new array/object), address changes → Angular notices and re-runs the pipe.

Example (Pure Pipe)

```
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({
  name: 'pureExample',
  pure: true // default
})
export class PureExamplePipe implements PipeTransform {
  transform(value: any[]): number {
    console.log('Pure pipe executed');
    return value.length;
  }
}
```

Behavior - If you push an item into the array (without reassigning it), the pipe won’t run because the reference didn’t change.

2. Impure Pipe

An Impure Pipe runs on every change detection cycle, whether or not the input changes. This can affect performance, but it's useful when you need to respond to mutable data.

When it runs

- On every change detection cycle.
- Even if input data is the same (useful for arrays/objects modified in place).

Example (Impure Pipe)

```
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({
  name: 'impureExample',
  pure: false
})
export class ImpureExamplePipe implements PipeTransform {
  transform(value: any[]): number {
    console.log('Impure pipe executed');
    return value.length;
  }
}
```

Behavior - If you push or pop elements into the same array, Angular re-runs this pipe every time.

Pure vs Impure Comparison Table

Feature	Pure Pipe	Impure Pipe
Execution	Only when input changes (by reference)	On every change detection
Performance	Fast	Slower
Use Case	Static or immutable data	Dynamic or mutable data
Default Behavior	pure: true	pure: false
Example	Date, UpperCase, Currency pipes	Custom pipes working with live or changing data

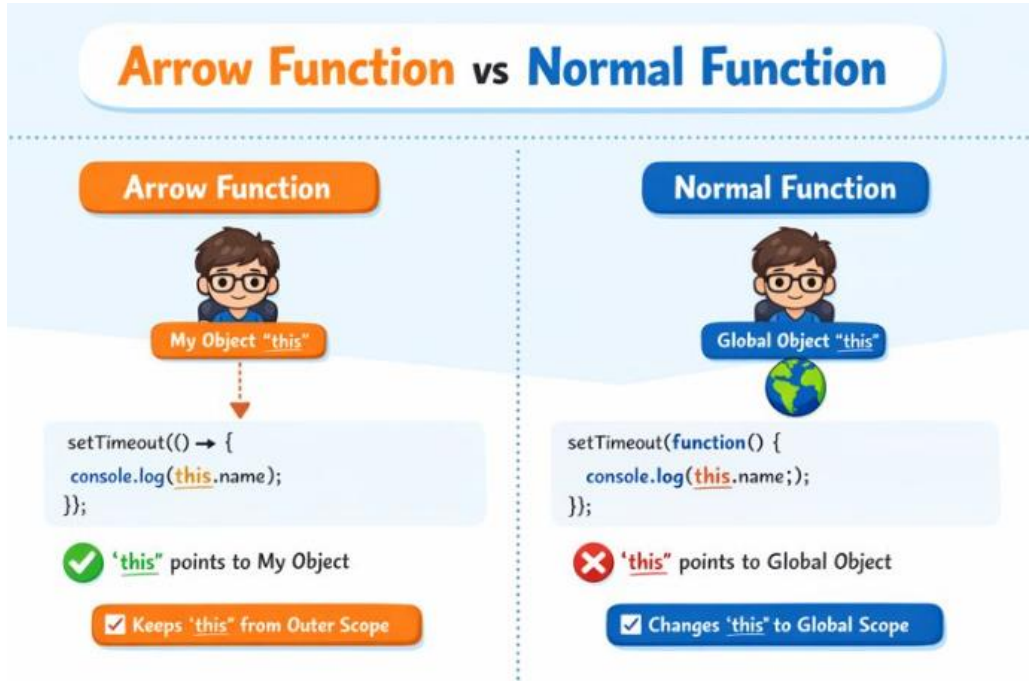
Q. What are Arrow Functions in Typescript?

Angular

Arrow Functions in TypeScript are a **shorter way to write functions** using the => syntax.

They are widely used in TypeScript/Angular/React because they are:

- Shorter
- Cleaner
- and automatically keep the correct this reference



Normal Function vs Arrow Function

✅ Normal Function

```
function add(a: number, b: number): number {
  return a + b;
}
console.log(add(10, 20)); // 30
```

✅ Arrow Function (short form)

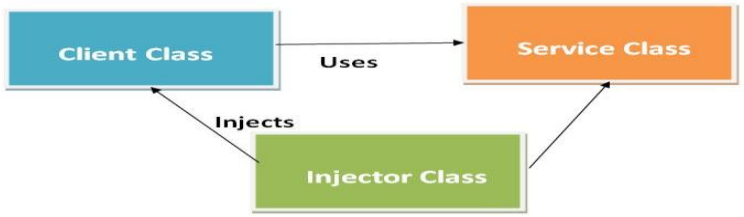
```
let add = (a: number, b: number): number => {
  return a + b;
};
console.log(add(10, 20)); // 30
```

Q. How to use Dependency Injector with Services in Angular?

Dependency Injection (DI) in Angular

👉 Angular automatically creates the Service object and gives it to the Component when needed.

Component does NOT create service using new keyword.
Angular injects it for you.



Real World Example

Example - Factory & Worker

- Service = Tool (Hammer)
- Component = Worker
- DI = Factory provides tool to worker

Worker doesn't build tool, factory gives it.

Same in Angular

Component doesn't create service, Angular provides it.

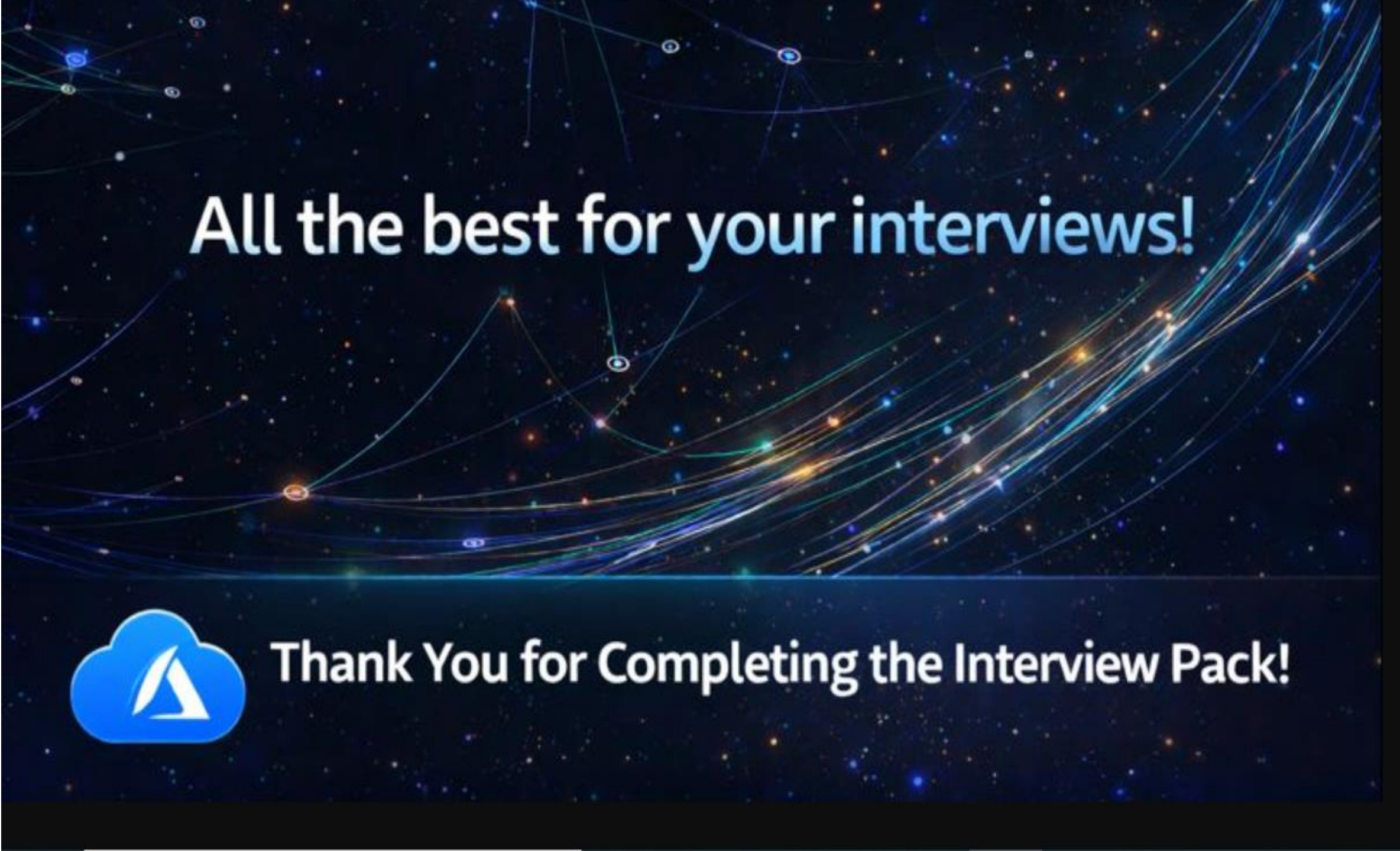
```
import { Component, OnInit } from '@angular/core';
import { ProductService } from './services/product.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent implements OnInit {

  message: string = '';

  // ✅ Dependency Injection happens here
  constructor(private productService: ProductService) {}

  ngOnInit(): void {
    this.message = this.productService.getMessage();
  }
}
```


All the best for your interviews!
Thank You for Completing the Interview Pack!