Angular TS – Asynchronization

[**✅ Task: Angular – Asynchronization with Observables & RxJS** 1](#_Toc205376062)

[**1️⃣ Purpose** 1](#_Toc205376063)

[**2️⃣ Theory** 1](#_Toc205376064)

[**3️⃣ Prerequisites** 1](#_Toc205376065)

[**4️⃣ Code Example (Step-by-Step)** 2](#_Toc205376066)

[**5️⃣ Project Structure Snapshot** 2](#_Toc205376067)

[**6️⃣ Summary** 3](#_Toc205376068)

**✅ Task: Angular – Asynchronization with Observables & RxJS**

**1️⃣ Purpose**

Understand the **core asynchronous behavior** in Angular using Observables and **RxJS** operators for real-time, reactive, and event-driven programming.

**2️⃣ Theory**

| **Concept** | **Description** |
| --- | --- |
| **Observable** | A stream of data you can subscribe to; can emit multiple values over time. |
| **Observer** | Object that defines callbacks: next, error, and complete. |
| **RxJS** | A library for reactive programming with Observables. Angular uses RxJS heavily. |
| **Operators** | RxJS provides operators like map, filter, debounceTime, switchMap, etc., to transform observable streams. |
| **Async Pipe** | Angular pipe to automatically subscribe to an Observable in the template. |

**3️⃣ Prerequisites**

* Angular CLI project created
* Basic understanding of components and services
* RxJS (comes by default with Angular)

**4️⃣ Code Example (Step-by-Step)**

**✅ Step 1: Create a Service that emits values (simulate data)**

ng g s services/timer

// src/app/services/timer.service.ts

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

import { Observable, interval } from 'rxjs';

import { map } from 'rxjs/operators';

@Injectable({

providedIn: 'root'

})

export class TimerService {

getCurrentTime(): Observable<string> {

return interval(1000).pipe(

map(() => new Date().toLocaleTimeString())

);

}

}

**✅ Step 2: Use the Observable in a Component**

ng g c components/clock

// src/app/components/clock/clock.component.ts

import { Component, OnInit } from '@angular/core';

import { TimerService } from 'src/app/services/timer.service';

import { Observable } from 'rxjs';

@Component({

selector: 'app-clock',

templateUrl: './clock.component.html'

})

export class ClockComponent implements OnInit {

time$!: Observable<string>;

constructor(private timerService: TimerService) {}

ngOnInit(): void {

this.time$ = this.timerService.getCurrentTime();

}

}

**✅ Step 3: Use AsyncPipe in Template**

<!-- src/app/components/clock/clock.component.html -->

<h3>Current Time: {{ time$ | async }}</h3>

**5️⃣ Project Structure Snapshot**

src/

│

├── app/

│ ├── services/

│ │ └── timer.service.ts

│ └── components/

│ └── clock/

│ ├── clock.component.ts

│ └── clock.component.html

**6️⃣ Summary**

* Angular uses **RxJS Observables** for asynchronous handling (like HTTP, timers, events).
* AsyncPipe helps you avoid manual subscriptions.
* RxJS operators like map, filter, debounceTime, mergeMap, switchMap, etc., are powerful tools for transforming and controlling streams.
* You can extend this to **search boxes**, **live counters**, **auto-complete**, **WebSockets**, etc.