In **RxJS**, the built-in `interval()` operator **waits** for the first period before emitting.

If you want **an immediate first emission** followed by regular intervals, you can combine `interval()` with `startWith()`.

Here's a complete Angular-friendly example:

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { interval, Subscription } from 'rxjs';
import { startWith } from 'rxjs/operators';

@Component({
  selector: 'app-timer',
  template: `<p>Value: {{ value }}</p>`
})
export class TimerComponent implements OnInit, OnDestroy {
  value = 0;
  private sub!: Subscription;

  ngOnInit(): void {
    // Emit immediately, then every 1 second
    this.sub = interval(1000)
      .pipe(
        startWith(0) // Emit 0 immediately before the first interval tick
      )
      .subscribe(count => {
        this.value = count;
        console.log('Tick:', count);
      });
  }
  ngOnDestroy(): void {
    // Prevent memory leaks
    if (this.sub) {
      this.sub.unsubscribe();
    }
  }
}
```

How it works:

- `interval(1000)` → emits `0, 1, 2, ...` every 1 second **starting after 1 second**.
- `startWith(0)` → forces an **immediate emission** before the first scheduled tick.
- The subscription is cleaned up in `ngOnDestroy` to avoid memory leaks.

✅ **Alternative**: If you want the emitted values to be **time-based** rather than count-based, you can use `timer(0, 1000)`:

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

timer(0, 1000).subscribe(val => console.log(val));
```

This starts at `0` immediately, then emits every second — no need for `startWith()`.

If you want, I can show you **a reusable Angular service** that provides an "immediate + interval" observable so you can inject it anywhere in your app.
Do you want me to prepare that?